НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«До захисту допущено»
В.О.Завідувача кафедри
_____ О.Л. Тимощук

Дипломна робота

на здобуття ступеня бакалавра
з напряму підготовки 6.040303 Системний аналіз

на тему: «Система планування та оптимізації графіку польоту екіпажів на основі методу Branch and Price»

Виконала:
студентка IV курсу, групи КА-63
Павлощук Ольга Олександрівна _____

Керівник:
канд. техн. наук, доцент
Тимощук Оксана Леонідівна _____

Консультант з економічного розділу:
доцент, к.е.н.
Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
Доцент, к.т.н. Коваленко А.Є _____

Рецензент:
_____
_____ _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В.О.Завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Павлощук Ользі Олександрівні

1. Тема роботи «Система планування та оптимізації графіку польоту екіпажів на основі методу Branch and Price», керівник роботи Тимощук Оксана Леонідівна, кандидат тех. наук, доцент, затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с__

2. Термін подання студентом роботи 08 травня 2020 року

3. Вихідні дані до роботи

4. Зміст роботи

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., к.е.н., доцент		
Нормоконтроль	Коваленко А.Є., доцент, к.т.н.		

7. Дата видачі завдання 13.04.2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	13.04.2020	Виконано
2	Збір інформації	27.04.2020	Виконано
3	Ознайомлення з літературою і підготовка теоретичної частини диплому	27.04.2020	Виконано
4	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	04.05.2020	Виконано
5	Розробка програмного продукту	11.05.2020	Виконано
6	Виконання функціонально-вартісного аналізу	28.05.2020	Виконано
7	Отримання висновків після тестування	29.05.2020	Виконано
8	Оформлення дипломної роботи	08.06.2020	
9	Отримання допуску до захисту та подача роботи до ДЕК	12.06.2020	

Студент Ольга ПАВЛОЩУК

Керівник Оксана ТИМОЩУК

РЕФЕРАТ

Бакалаврська робота: 86 с., 20 рис., 8 табл., 2 додатки, 24 джерел.

МЕТОД ГІЛОК ТА ЦІНИ, ЗАДАЧА МАРШРУТИЗАЦІЇ, МЕТОД ГІЛОК ТА МЕЖ, ПЛАНУВАННЯ ГРАФІКУ, ТРАНСПОРТИЗАЦІЯ.

Актуальність дослідження зумовлена необхідністю пізнання еволюції становлення сучасної авіації, механізмів її діяльності та створенням алгоритмів удосконалення ефективності функціонування відповідно до нагальних потреб сьогодення.

Мета роботи – дослідження історії та окреслення еволюції існуючих методів транспортної маршрутизації; визначення чинників динаміки алгоритму Branch and Price; створення комп'ютерної моделі запропонованих змін в удосконаленому алгоритмі транспортної маршрутизації.

Об'єкт дослідження – алгоритм оптимізації графіку польоту екіпажів на основі методу Branch and Price.

Методи дослідження – специфіка об'єкту і поставлені завдання зумовили застосування загальнонаукових (аналіз, синтакс) і спеціальних методів і прийомів (описовий, системно-класифікаційний, функційного аналізу, прийом кількісних підрахунків, прийом трансформаційного аналізу).

Наукова новизна полягає в тому, що вперше досліджено алгоритм Branch and Price як системний динамічний феномен, розроблено і застосовано для його пізнання комплекс прийомів, окреслено механізми, джерела та чинники розвитку сучасних авіаперевезень.

Результатом роботи є розроблений програмний продукт для побудови графіку польотів суден цивільної авіації.

Шляхи подальшого розвитку об'єкту дослідження – методи оптимізації графіків польоту за допомогою більш сучасних методів та підходів.

ABSTRACT

Diploma work: 86 p., 20 fig., 8 tabl., 2 appendixes, 24 sources.

BRANCH AND PRICE, ROSTERING PROBLEM, BRANCH AND BOUND, SCHEDULE PLANNING, TRANSPORTATION.

The relevance of the study is due to the need to know the evolution of modern aviation, the mechanisms of its activities, and the creation of algorithms to improve the efficiency of functioning by the urgent needs of today.

The purpose of the work is to study the history and outline the development of existing methods of transport routing; determination of dynamics factors of the Branch and Price algorithm; creation of a computer model of the proposed changes in the advanced transport routing algorithm.

The object of study - an algorithm for optimizing the flight schedule of crews based on the method of Branch and Price.

Research methods - the specifics of the object and the tasks led to the use of general scientific (analysis, syntax) and special methods and techniques (descriptive, system-classification, functional analysis, reception of quantitative calculations, reception of transformational analysis).

The scientific novelty is that for the first time the Branch and Price algorithm as a system dynamic phenomenon was studied, a set of techniques was developed and applied for its knowledge, the mechanisms, sources, and factors of development of modern air transportation were outlined.

The result of the work is a developed software product for scheduling flights of civil aircraft.

Ways of further development of the object of study - methods of optimizing flight schedules using more modern methods and approaches.

ЗМІСТ

ВСТУП

Вплив та значення авіації в сучасному світі важко переоцінити. Авіаіндустрія займає чи не найголовнішу роль в міжконтинентальному транспортному сполученні, що й робить її такою життєвонеобхідною для розвитку глобальної економічної торгівлі та міжнародного туризму. Тільки за попередніми оцінками в 2019 р. було здійснено близько 39 мільйонів авіарейсів [1]. В Україні цей показник досягнув позначки в 335 тисяч рейсів, що на 10% більше, ніж за попередній рік. За даними Державного підприємства обслуговування повітряного руху України (УкрАероРух) у трійку лідерів компаній-перевізників увійшли МАУ, Turkish Airlines та WizzAir, більш детально зображено на Рисунку 1 [2].

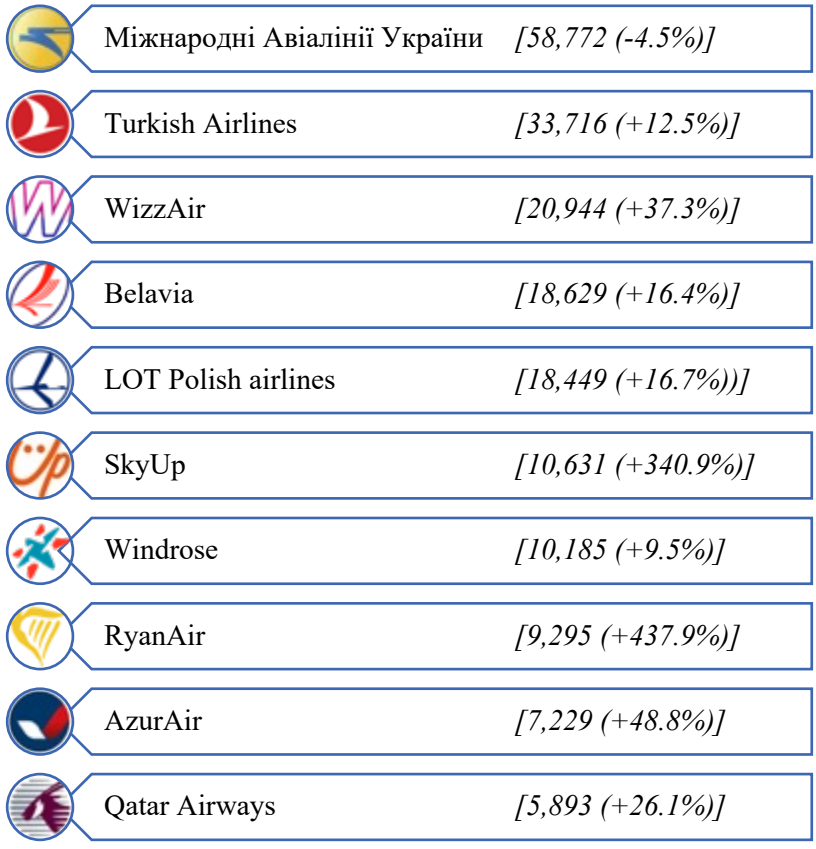


Рисунок 1 – Топ-10 авіаперевізників України (2019 р.)

Для забезпечення безперебійного польоту кожного з рейсів та всієї індустрії в цілому до роботи щодня залучається 10.2 мільйони висококваліфікованого персоналу, з яких безпосередньо на авіалініях працює 2.7 мільйони, докладніше варто дивитися на Рисунку 2 [3].

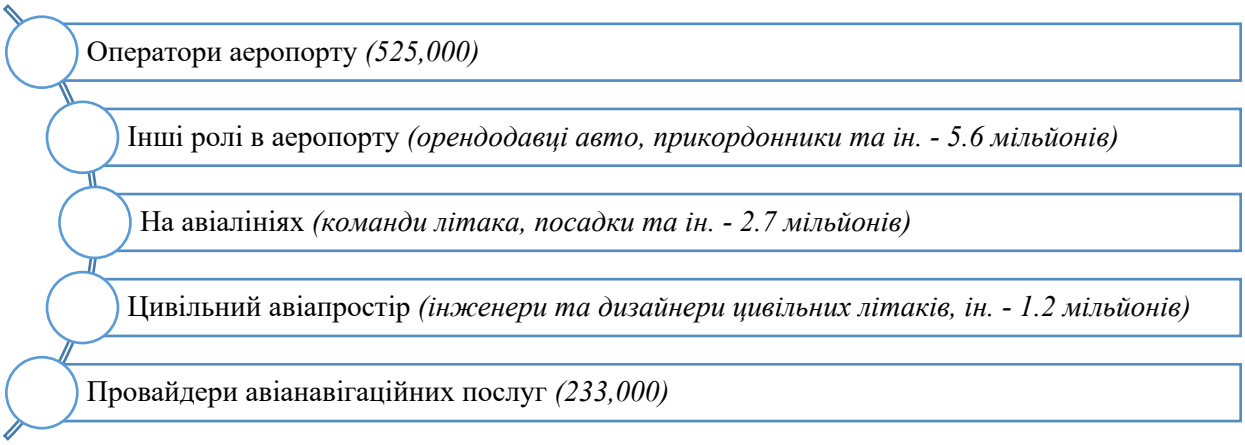


Рисунок 2 – Зайнятість в авіаіндустрії

В авіаіндустрії видатки на працю є другими за величиною після витрат на паливо [4]. Затрати на працю фіксуються в основному в короткостроковій перспективі, у той час як на пальне можуть сильно коливатися в залежності від біржової ціни на нафту. Хоча видатки на працю займають близько 35% від загальних експлуатаційних витрат авіакомпанії, які у свою чергу становлять 75% від усіх операційних витрат, аналітики втім продовжують акцентувати левову частку своєї уваги виключно затратам на пальне [5].

Турбулентний розвиток попиту на послуги авіаіндустрії призвів до того, що компанії перебувають у постійній конкуренції між собою. Специфіка самого бізнесу спричинила боротьбу за критично не лояльного клієнта, який зосереджує свою увагу, як правило, на ціні, а не на якості сервісу. Тож під час економічних спадів керівництво прагне скоротити витрати в першу чергу саме на оплаті праці, зменшивши кількість працівників чи урізавши розмір ставки заробітної плати.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія повітряних перевезень у світі

Історії відомо вже більше двох тисяч років різноманітних спроб людини піднятися в повітря, від повітряних зміїв та стрибків з веж до польотів в сучасній турбоері. Під терміном *повітряні перевезення* зазвичай науковці мають на увазі певні пересування пасажирів або вантажів будь-яким транспортним засобом, що може підтримувати керований політ.

З моменту першого комерційного пасажира в 1914 р. відбулися фундаментальні зрушення в авіаційних перевезеннях. На вдосконалення літальних апаратів та підготовку досвідчених кадрів чи не найголовнішим чином вплинули Перша та Друга світові війни. Тісний зв'язок авіації та військової справи на початку Першої світової заклав міцний фундамент для прискореного розвитку комерційної авіації, навіть по закінченню трагічних подій. На початку Другої світової війни вже був здійснений перший політ Нью-Йорк – Чикаго, що ознаменував початок нової ери швидкозмінних технологій. Друга Світова війна сприяла популяризації авіасполучення, адже саме літаки стали одним із ключових елементів воєнних тактик та стратегій тих років. Це спричинило появу великої кількості нових аеропортів та висококваліфікованих пілотів. Також відбувся стрімкий прорив в реактивний авіації та навігаційній допомозі в орієнтуванні, що сприяло авіації зайняти основну транспортну нішу після війни.

У той же час левову частку у розвитку авіаіндустрії відіграли введення необхідних регуляторних норм та урядове адміністрування в транспортній галузі. Зокрема, європейське законодавство сприяло запуску нових авіаліній, а по той бік Атлантики, уряд США почав потужне субсидіювання авіапошти,

що і через десятиліття залишатиметься одним із найголовніших джерел комерційної вигоди для авіакомпанії.

Важливим етапом у розвитку авіації стало популяризація використання реактивних двигунів, що значно збільшили швидкість судна і, як наслідок, скоротили час польоту. На Рисунку 1.1 наведено порівняння часу польоту від Нью-Йорка на літаках з різними типами двигунів.

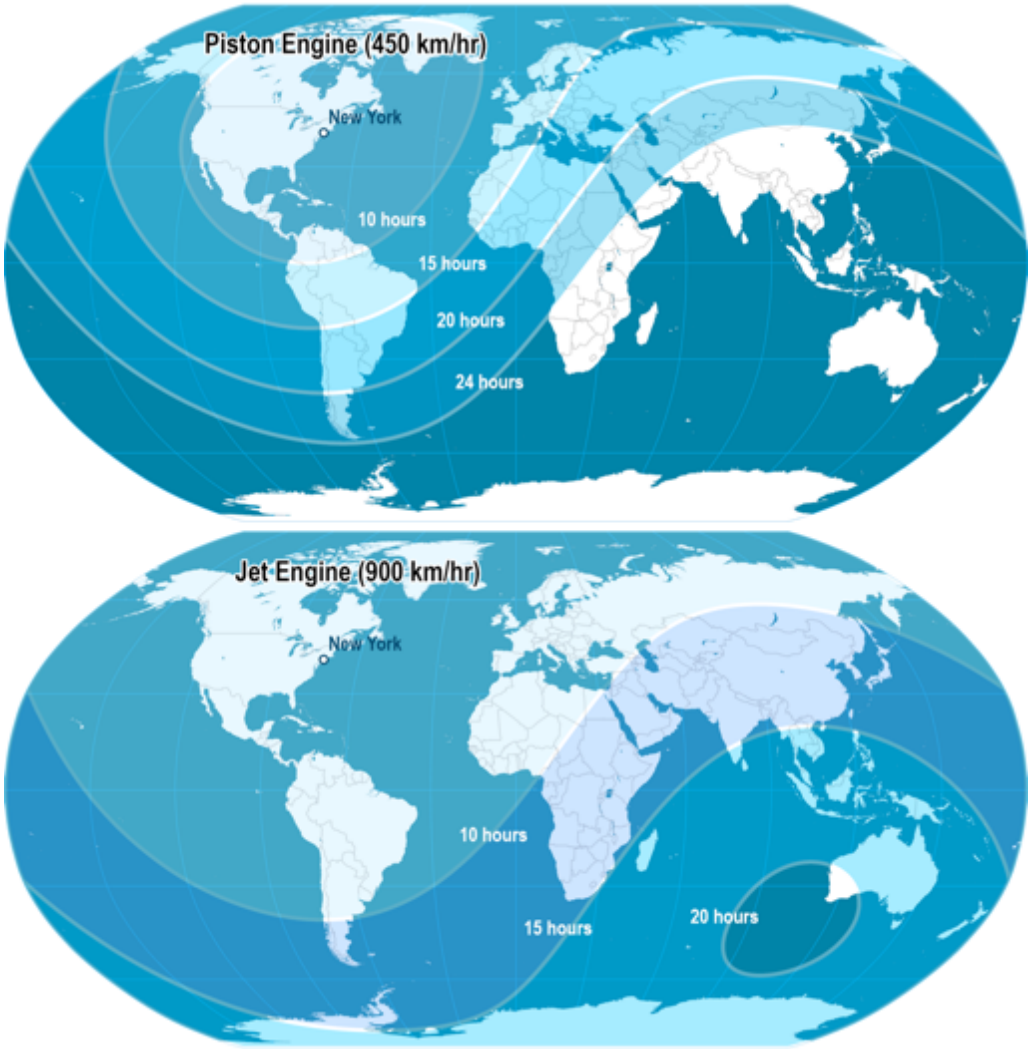


Рисунок 1.1 – Порівняння часу польоту від Нью-Йорка на поршневому (зверху) та реактивному (знизу) двигунах

Авіаперевезення на сьогодні є найбільш домінантним стилем подорожування на міжконтинентальні напрямки і стали надзвичайно конкурентно спроможними в перельотах на короткі дистанції на багатьох регіональних ринках.

Найвизначнішим досягненням сучасної авіації заслужено варто вважати зменшення кількості ризиків та інцидентів під час польоту. Яскравим прикладом цієї думки слугує таке співвідношення: якби цивільна авіація мала такий же показник нещасних випадків, як у 1960-х, то у 2018 р. *щодня* відбувалося б по три інциденти, у той час, як було зафіксовано, до дев'яти *за увесь рік* [6].

Важливим для світової торгівлі є перевезення літаками вантажів. За оцінками експертів частка таких транспортувань у міжнародній економіці становить менше одного відсотка, та за вартістю такі товари займають 35% від загальної ціни перевезених вантажів [6]. Повітряні перевезення є як ніколи актуальними для групи товарів, що необхідно швидко перемістити на великі відстані за короткий проміжок часу, або в тому випадку, коли швидкість доставки є важливішою, за потенційні економічні витрати.

Цивільна авіація має яскраву історію та ще довший список досягнень у впливі на сучасний світ у найрізноманітніших сферах життєдіяльності людини, як з позитивної точки зору (розвиток туризму та поширення культурного різноманіття), так і з негативного боку (екологічне забруднення та поширення інфекційних захворювань).

1.2 Географія мереж повітряних шляхів

Теоретично повітряні судна мають значно більшу свободу у виборі маршруту, ніж інші види транспорту, тому авіалінії охоплюють океани, гірські ланцюги, важкодоступні пустелі та інші фізичні бар'єри. Основні санкції, що накладаються на польоти були закладені в Чиказькій конвенції від 1944 р. та передбачають дев'ять Свобод повітряного простору (Freedoms of the air, див. детальніше Таблицю 1.1 та Рисунок 1.2).

Таблиця 1.1 – Свободи повітряного простору

№	Опис	Приклад
I	Проліт над іноземною країну (без посадки).	З Мексики до Канади мексиканською авіакомпанією над США.
II	Дозаправка (або технічне обслуговування) без висадки пасажирів.	Дозаправка в Ірландії при польоті з з Великої Британії в США британською авіакомпанією.
III	З "домашньої" країни в іншу.	Політ австралійською авіакомпанією з дому в Китай.
IV	Політ з "третьої" країни додому.	Політ з Чилі бразильців додому.
V	Політ між двома іноземними країнами в рейсі, що починається чи завершується власною (-ій).	Рейс з ОАЕ в Нову Зеландію, здійснений авіакомпанією Emirates (ОАЕ) із зупинкою в Австралії.
VI	Право на політ з однієї іноземної країни до іншої та зупинкою у власній з нетехнічних причин.	Політ з Австралії до США французькою авіакомпанією із зупинкою у Французькій Полінезії.
VII	Переліт між двома іноземними країнами без "заходу" до власної.	З Португалії до Німеччини ірландською авіакомпанією.
VIII	Політ в іноземній країні з продовженням у власній.	Рейс англійської авіакомпанії між Сан-Франциско та Лондоном, з повною зупинкою в Нью-Йорку.
IX	Право літати в іноземній країні без продовження рейсу у власній.	Переліт з Лондона в Стамбул італійською авіакомпанією.

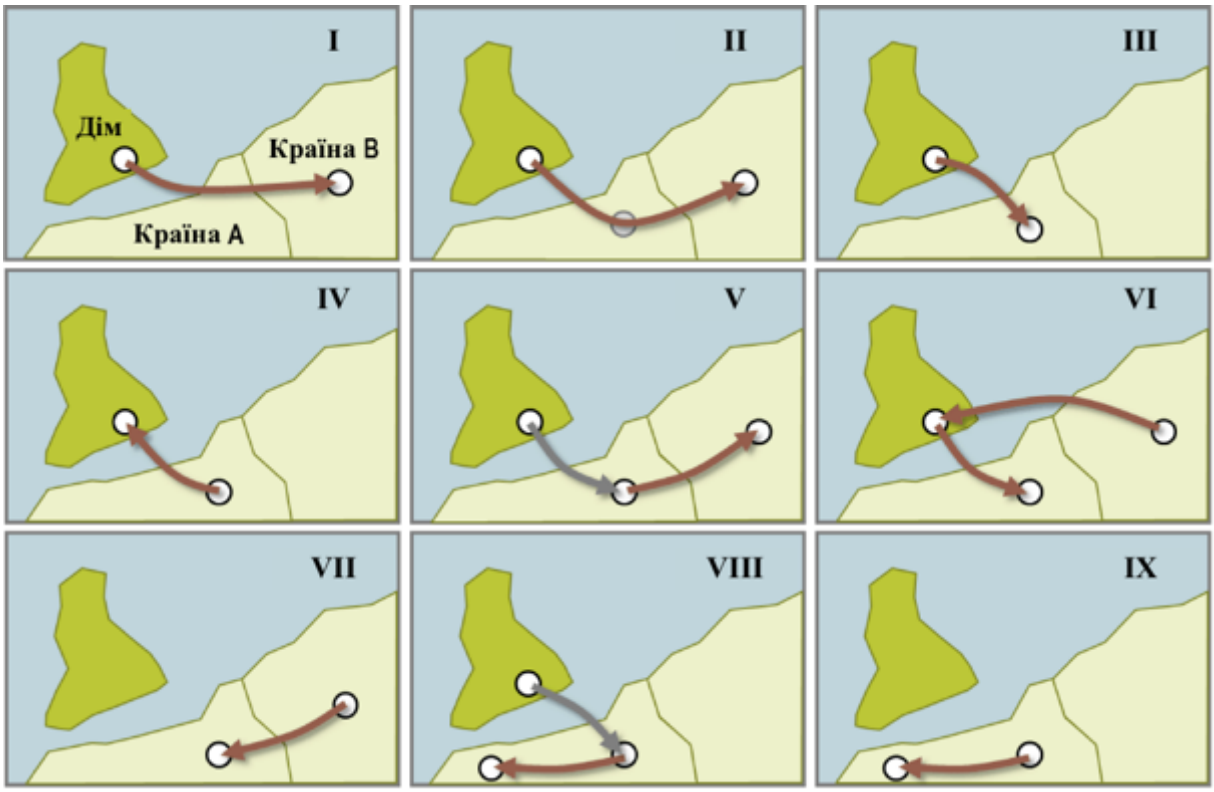


Рисунок 1.2 – Ілюстрація конвенції Свобод повітряного простору

Важливою складовою авіаперевезень стала поява окремих вантажних служб повітряного транспорту, які зазвичай оперують у відмінних від пасажирських мережах. Це пояснюється перш за все тим, що пасажирські мережі орієнтовані на мандрівників, а не на вантажовідправників, а це у свою чергу дало поштовх до появи операцій з легшими літаками, що мають більшу вантажомісткість. В індустрії досить поширеними є комбіновані перевізники (наприклад, Lufthansa), які здійснюють свою діяльність як на ринках пасажирських, так і вантажних перевезень, та мають у розпорядженні змішаний флот літаків.

Цікавою особливістю мереж авіаліній, про яку варто згадати, є їх сезонність. Повітряні вантажні потоки досягають свого піку напередодні різдвяного сезону, хоча деякі конкретні продукти (наприклад, квіти до Дня святого Валентина) мають іншу часову структуру. Для пасажирських повітряних перевезень піковими місяцями за статистикою є липень та серпень, що відповідають апогею туристичного сезону в Європі та Північній Америці. В інших світових регіонах більш важливими можуть бути інші сезонні моделі,

як-от, у Китаї найпотужніші дні року, як правило, близькі до Весняного фестивалю (або Місячного Нового року) у січні чи в лютому.

1.3 Хаби аеропортів та альянси авіакомпаній

Велика кількість авіакомпаній мають досить централізовані мережі, а домашні аеропорти найбільших гравців ринку – одні з найзавантаженіших у світі. Головна ідея системи з хабами полягає у використанні проміжного вузла аеропорта (див. докладніше Рисунок 1.3). Саме на цьому етапі і виникають перші труднощі в збереженні цілісності розкладу, адже більшість пасажирів використовують сполучні рейси.

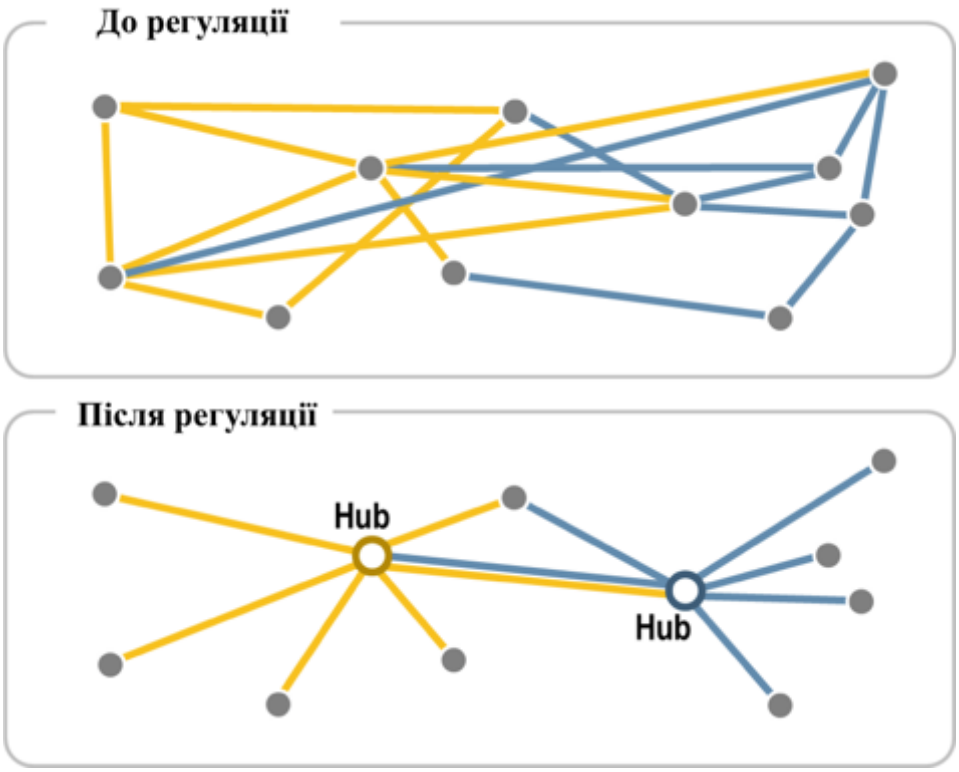


Рисунок 1.3 – До та після введення регуляторного хабу

Тривалий час поширенню авіакомпаній на міжнародному рівні перешкоджали обмеженість постійним регуляторними нормами від країни, що приймає рейс, та культурна спорідненість мандрівників з національними перевізниками. Ці труднощі були подолані компаніями за рахунок утворення альянсів – добровільних угод для посилення конкурентних позицій партнерів. Компанії залишаються комерційно незалежними і в той же час виграють за рахунок зниження трансакційних витрат та рівномірного розподілу ризиків. На сьогодні існує три основних повітряних альянси:

- Star Alliance (27 компаній, серед яких Air Canada, Lufthansa та United Airlines);
- SkyTeam (19 компаній на чолі з Delta і Air France);
- Oneworld (13 компаній, очолювані British Airways та American Airlines).

Серед основних переваг утворення альянсів можна виділити такі:

- Обмін кодами. Члени альянсу можуть продавати місця на рейсах один одного.
- Оптимізація з'єднань. Яскравим прикладом слугують суміжні ворота в спільних терміналах, що прискорюють зв'язки, наприклад, термінал 3 Міжнародного аеропорту Пекіна, де всі авіакомпанії Star Alliance, які обслуговують місто, розташовані спільно.
- Географічна спеціалізація. Авіакомпанія в альянсі може використовувати світові ринки, спеціалізуючись виключно на своєму внутрішньому. Наприклад, член альянсу Star Air Canada обслуговує лише сім вузлів у Східній та Південно-Східній Азії, але завдяки своїм партнерам по альянсу отримує доступ до десятків інших міст регіону. У свою чергу, азіатські члени Star Alliance, такі як Singapore Airlines (SIA), отримують доступ до великої мережі Air Canada, без жодного власного рейсу до Канади, у випадку SIA.
- Спільний маркетинг. Зокрема, накопичення миль у межах альянсу.

Хоча минуле століття стало свідком різкого зростання кількості авіаперевезень, важливі неподолані виклики все ще затьмарюють їх майбутнє. До задач, які постають перед галуззю перш за все варто віднести боротьбу з тероризмом та вдосконалення систем безпеки. Не менш важливим також є початок використання екологічно чистого пального в перевезеннях та оптимізації графіку польотів, у зв'язку зі збільшенням навантаження на лінії.

1.4 Графік польоту: етапи формування

Щомиті в небі перебуває в середньому до 11 тисяч пасажирських лайнерів [7], і чималу роль в ефективній оптимізації мереж повітряного простору відіграє правильно розроблений графік польотів. Загальний процес його формування зображено на Рисунку 1.4.

Рисунок 1.4 – Процес формування графіку польоту.

Головною метою кожного етапу послідовності є пошук рішення для застосування на наступному кроці процесу [8].

Першим пунктом в формуванні графіка польотів є складання загального розкладу. На цьому етапі авіакомпанії зазвичай беруть до уваги декілька основних факторів, що впливатимуть на розрахунок ймовірнісних характеристик своєї економічної діяльності, а саме:

- прогнозування споживчого попиту на потенційний продукт (рейси);
- наявна у розпорядженні інформація про доступність власних ресурсів.

В результаті коректної роботи першого кроку буде отримано набір рейсів, які авіаперевізник готовий здійснити в рамках своїх потенційних потреб та реальних можливостей.

Наступним етапом у формуванні графіка польотів слугує призначення літаків на заплановані рейси. На коректне виконання рейсу в першу чергу впливають технічні параметри авіалайнера (конструкція фюзеляжу, кількості пасажирів, вантажопідйомності тощо). Цей крок має на меті визначити кількість літаків та їх конкретні моделі, які будуть застосовані компаніями на заплановані уже на попередньому кроці рейси.

Фіксований список рейсів з означеними повітряними суднами дає змогу почати роботу над наступним кроком планування, а саме, маршрутизацією літаків і розробкою розкладу для екіпажів. Саме на цьому кроці виникають перші труднощі, пов’язані з великою кількістю регламентацій роботи членів бортового судна (переважна більшість з них керуються трудовим законодавством країни, в якій розташована авіакомпанія, та міжнародними регуляторними актами) та використанням тієї чи іншої моделі літака [9].

Дана робота розглядає останній етап створення розкладу. Планування графіку робіт моделюється як детермінована задача цілочисельного лінійного програмування, мета якого є мінімізація витрат на оплату робочого часу персоналу на борту літака [10], [11]. Авіакомпанії вигідно скорочувати частку оплачуваного часу, що не використовується безпосередньо для виконання польотів [12], [13].

1.5 Дослідження методів розв'язку задачі побудови розкладу

Задача побудови розкладу для екіпажів авіакомпанії є об'єктом наукових досліджень уже протягом тривалого проміжку часу. Вона завжди була тісно пов'язана з транспортною маршрутизацією. Один з перших методів для

розв'язку завдання про складання графіка робіт для бортового екіпажу був описаний в 1968 р. У дослідженні були наведені евристичні підходи, адже в той час не було можливості отримати точне рішення через відсутність комп'ютерів з необхідною обчислювальною потужністю [14].

Уже кілька років потому була опублікована робота, в якій були описані методи, що дозволяли автоматизувати процес планування [15], але через всю ту ж обмеженість наявних у розпорядженні ресурсів обчислювальних машин не було змоги відмовитися від евристичних методів.

Зі стрімким розвитком електронно-обчислювальної техніки було створено багато точних алгоритмів для пошуку оптимального розв'язку. У 2004 р. командою вчених була розроблена система прийняття рішень для планування робіт екіпажів, заснована на методі Branch and Price [16]. Branch and Price являє собою метод комбінаторної оптимізації для вирішення цілочисельної задачі лінійного програмування з великою кількістю змінних [17]. Відома дослідниця у сфері транспортної логістики Синтія Барнхарт разом з командою опублікували роботу про використання методу Branch and Price в даній галузі [18].

Алгоритм Branch and Price поєднує в собі метод гілок і меж та метод генерації стовпців для матриці задачі [19]. Метод гілок і меж вперше був запропонований в 1960 році та включає в себе три етапи: розбиття на гілки, вибір гілки і обчислення меж [20]. Основна перевага методу полягає в відсіканні гілок, де обчислена межа гірше, ніж поточне краще рішення.

Для розв'язку задачі з великою кількістю змінних використовується метод генерації стовпців. Алгоритм можна розділити на дві частини: основна та допоміжна задача (підзадача). Мета основної задачі – це знаходження допустимого розв'язку для підмножини змінних вихідної задачі. Для переходу до підзадачі необхідно провести релаксацію і знайти рішення двоїстої задачі. Допоміжна задача полягає в пошуку нових змінних, які не були включені в основну задачу. Цільова функція підзадачі залежить від поточних значень змінних. Завдання пошуку нової змінної для основної задачі еквівалентна

задачі пошуку обмеження, що частіше порушується, в двоїстому завданні. У методі Branch and Price допоміжна задача здійснює пошук цього обмеження, щоб забезпечити виконання більшої кількості обмежень в двоїстій задачі. Додавання нової змінної в основну задачу призводить до зміни двоїстих змінних, які впливають на цільову функцію підзадачі.

Метод генерації стовпців виконується до тих пір, поки всі обмеження двоїстої задачі не будуть виконані. Це буде у свій час означати, що оптимальне значення релаксованої задачі знайдено [21]. Якщо знайдений розв'язок цілий, то вихідна задача буде вважатися розв'язаною, в іншому випадку необхідно буде виконати розгалуження по дробовим змінним [22].

Висновки до Розділу 1

З моменту перших польоту братів Райт в 1903 р. минуло вже більше 110 років. За цей порівняно короткий проміжок часу авіація змогла зробити колосальні зрушення в різних напрямках розвитку.

Ключову роль в цих зрушеннях відіграли Перша та Друга світові війни. Ці трагічні події дали поштовх як у сфері технічного прогресу (особливо яскраво це було помітно під час Великої Вітчизняної, коли літальні апарати були незамінними у воєнних стратегіях та тактиках), так і в підготовці висококваліфікованого персоналу, який в переважній більшості залишився незайнятим після завершення бойових дій. Стрімкий розвиток авіації дав можливість здійснення трансконтинентальних повітряних перевезень, що сприяло розвитку світової торгівлі, міжкультурній взаємодії та розвитку туристичної галузі економіки.

Наукові досягнення, основу яким було закладено ще за часів війни, дали змогу застосувати навички реактивного будування двигунів в цивільному житті, а розробка нових методів навігаційного координування, зокрема

використання радіохвиль, стало апогеєм застосування та розвитку повоєнних технологій на практиці авіаіндустрії.

Авіаперевезення пасажирів та вантажу за багатьма показниками є найефективнішим та найбільш оптимальним способом пересування. Таких рівнів відповідності останнім регуляторним нормам та законам міжнародної авіації в першу чергу вдалося досягти за рахунок вигідної для розвитку цієї галузі урядовим рішенням по обидва береги Атлантики.

Цивільна авіація ні на секунду не припиняє свого розвитку. Насамперед це відчутно в питаннях безпеки та боротьбі з терористичними загрозами сучасності. Виклики, пов'язані із захистом навколишнього середовища, теж відіграють не останню роль в планах на майбутнє авіаіндустрії.

Розвиток сучасної авіації досяг нечуваного донині розмаху, оскільки повітряні перевезення не заангажовані територіальними перешкодами. Основні обмеження, що накладаються на повітряний транспорт та їх взаємодію під час польоту та з транзитними країнами, були створені людиною, а найголовнішими з них є Свободи повітряного транспорту, затверджені Чиказькою конвенцією (1944 р.).

Невід'ємною складовою перевезень в авіаіндустрії стала поява окремих вантажних служб авіатранспорту, які зазвичай оперують у мережах, відмінних від пасажирських. Проте існують оператори, які задовольняють як потреби мандрівників, так і власників вантажу, і мають у своєму розпорядженні флот суден для обох видів перевезень.

Сезонність в авіаперевезеннях теж відіграє значну роль в плануванні графіку польотів. Наприклад, піком туристичного сезону в Європі та Північній Америці вважають липень та серпень, а для вантажних перевезень часова структура зміщується на грудень та січень.

Значну роль в організації сучасної авіаційної індустрії відіграють домашні аеропорти найбільших перевізників, які зазвичай функціонують як сполучні вузли. Саме в розробці коректного графіку польотів між хабами і полягає провідне завдання спеціалістів з транспортної маршрутизації.

Повітряні альянси, членами яких є 50+ найбільших авіакомпаній, задовольняють потреби своїх клієнтів навіть на тих ринках, де окрема компанія слабо представлена або не оперує взагалі. До основних переваг утворення таких об'єднань зокрема ще входить обмін кодами, оптимізація з'єднань, географічна спеціалізація та спільний маркетинг.

У коректній взаємодії 11 тисяч пасажирських літаків, що щомиті перебувають в небі, чималу роль відіграє правильно спланований графік польотів. Загальний його процес складається з чотирьох основних етапів:

- створення розкладу польотів;
- призначення борту;
- маршрутизації;
- створення графіку для роботи вже відповідного екіпажу.

Кожен з цих етапів не може бути пропущеним чи замінений, хоча деякі методи пропонують паралельне опрацювання останніх двох кроків задля пришвидшення роботи. Дана робота розглядає у свою чергу саме останній етап створення розкладу, який моделюється як детермінована задача лінійного програмування, за мету якого покладено мінімізацію витрат авіакомпанії на заробітні плати членів екіпажу літака.

Хоча дана задача тісно пов'язана із питанням транспортної маршрутизації, лише останні 50 років їй приділяється належна увага. Перші розробки та запропоновані методи були в більшості своїй обмежені потужностями тодішніх електронно-обчислювальних машин. Та ця перепона з часом була подолана і вже у 2004 р. був запропонований метод Branch and Price, удосконалення якого у відповідності з потужностями сучасної техніки і стало однією з основних цілей дослідження.

РОЗДІЛ 2 МАТЕМАТИЧНА МОДЕЛЬ

2.1 Постановка задачі

Задачу формування розкладу для бортового екіпажу моделюватимемо як задачу про розбиття. Під маршрутом матимемо на увазі певний циклічний набір сусідніх рейсів, які члени екіпажу встигають пролетіти за одну зміну.

Тоді задача може бути сформульована наступним чином [10]:

$$\sum_{r \in R} c_r x_r \to min$$

$$\sum_{r \in R} a_{ir} x_r = 1, \forall i \in I^*$$

$$x_r \in [0,1],$$

де $r \in R$, а R – множина всіх можливих маршрутів для даного розкладу рейсів з назначеним флотом. I – множина всіх рейсів, а $I^* \subset I$ – лише обов'язкові рейси.

$$a_{ir} = \begin{cases} 1, & \text{якщо } i \in r \\ 0, & \text{в іншому випадку} \end{cases}$$

$$x_r = \begin{cases} 1, & \text{якщо } r \text{ входить в розв}'\text{язок} \\ 0, & \text{в іншому випадку} \end{cases}$$

$$c_r = \sum_{r \in R} c_i - \text{вартість маршруту } r$$

Запропоноване в роботі формулювання також може бути послаблене, якщо дозволити виконувати обов'язкові рейси повторно. Обов'язковими будемо називати рейси, що є економічно вигідними для авіакомпанії, в той час, як необов'язковими – ті, виконання яких не приносять суттєвого економічного зиску.

2.2 Метод Branch and Price

Branch and Price – це модифікований метод гілок і меж. Для пошуку розв'язку задачі лінійного програмування з великою кількістю змінних застосовується метод генерації стовпців [23]. На Рисунку 2.1 окреслено основні етапи алгоритму.

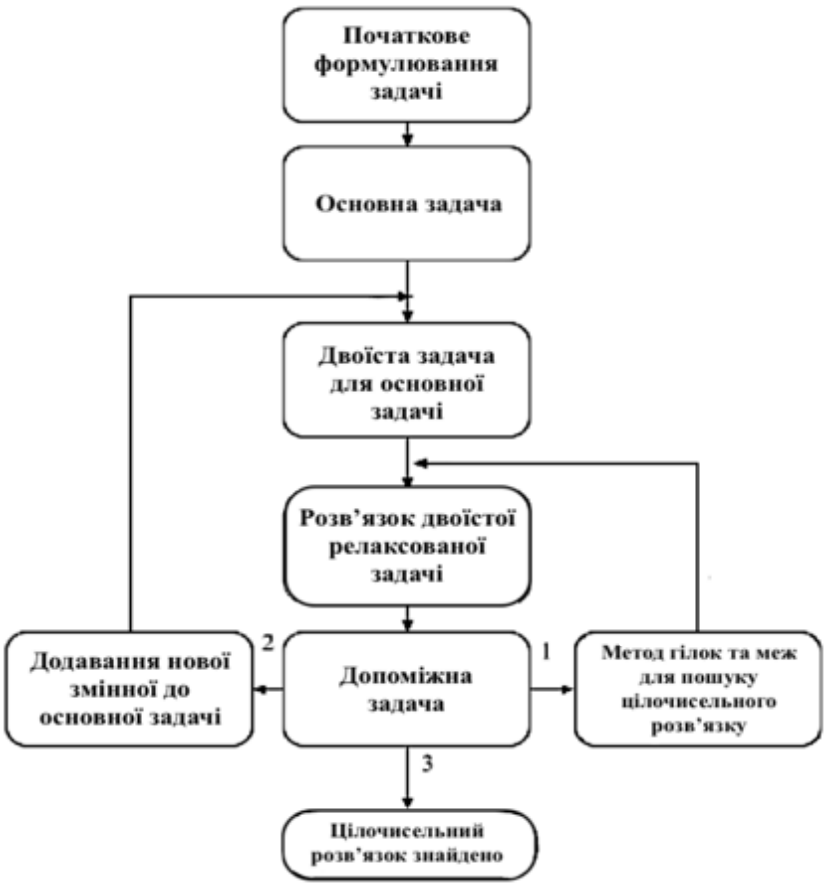


Рисунок 2.1 – Основні етапи алгоритму Branch and Price

На етапі пошуку рішення допоміжної задачі алгоритм має три гілки продовження:

1. Якщо всі обмеження двоїстої завдання виконані, але розв'язок основного завдання є не цілим.
2. Якщо обмеження двоїстої задачі не виконане.
3. Якщо всі обмеження двоїстої задачі виконані і розв'язок основної задачі цілий. У цьому випадку початкова задача вважається розв'язаною.

2.2.1. Евристичний алгоритм для побудови основної задачі

У цій частині роботі наведено опис евристичного алгоритму для пошуку початкової підмножини змінних, яка ввійде в основну задачу. Реалізований алгоритм має на меті знаходження набору допустимих маршрутів, який покриватиме всі обов'язкові рейси з мінімальною вартістю.

Базисна процедура алгоритму повертає підмножину змінних (набір маршрутів), для якого будемо розв'язувати основне завдання.

Ми отримуємо рішення шляхом послідовного додавання маршруту до загальної множини маршрутів. У свою чергу кожен маршрут також отримується послідовним додаванням рейсів. Основна мета – це покриття всіх обов'язкових рейсів циклічними маршрутами (*цикл* – набор сусідніх рейсів, у яких перший та останній аеропорти співпадають), тому кожен маршрут повинен включати в себе якомога більше польотів, в яких зацікавлена авіакомпанія.

Список обов'язкових рейсів складається з авіарейсів з різною вартістю. Основна ідея алгоритму – знайти найдорожчий елемент з даної підмножини і покрити його в першу чергу.

Одним з наступних кроків здійснюється пошук циклічних маршрутів, тобто перший і останній аеропорти в зміні для екіпажу повинні бути

однаковими. В першу чергу необхідно забезпечити побудову саме замкнутого набору рейсів.

Провідною особливістю алгоритму можна вважати першочергове покриття всіх обов'язкових рейсів та намагання завершити виконання без залишку непокритих рейсів.

Отриманий набір R дозволяє сформулювати основну задачу. Відомо, що значення цільової функції релаксованої задачі і двоїстої до неї збігаються в оптимальних точках. Релаксація завдання дозволяє полегшити пошук рішення (обмеження $x_r \in [0,1]$ замінюється на $0 \le x_r \le 1$). Двоїсту задачу у свою чергу можна сформулювати наступним чином:

$$\sum_{i \in I} y_i \to max$$

$$\sum_{i \in I} a_{ir} y_i \le c_r, \forall r \in R$$

$$y_i \ge 0$$

У двоїстої задачі вже експоненційна кількість обмежень. Для її розв'язку спочатку розглядається лише підмножина обмежень. Допоміжна задача методу Branch and Price дозволяє знайти обмеження для двоїстої задачі, що найчастіше порушується, і доповнити основне завдання новою змінною, яка відповідає цьому обмеження.

2.3 Допоміжна задача

Мета допоміжної задачі – знайти саме порушене обмеження для двоїстої задачі. Отримані значення двоїстих змінних використовуються в цільової функції допоміжної завдання, яка формується наступним чином з обмежень пов'язаного завдання.

$$\sum_{j \in r} (c_j - y_j^*) \to min$$

В даному випадку вважаємо параметр y_j^* розв'язком двоїстої задачі, а c_j – вартістю j-ого рейсу. Значення $y_j^* = 0$, якщо $j \notin I^*$.

Обмеження вважається знайденим, якщо цільова функція підзадачі набуває від'ємного значення. Для пошуку використовується метод гілок і меж. Реалізований евристичний алгоритм здійснює пошук початкового рішення, що оптимізує роботу Branch and Bound (гілок та меж). Знайдене обмеження додається до основної задачі, в якості змінної. Потім релаксована задача розв'язується заново.

У роботі використовуються дані за трьохденний період (тобто будь-який запланований маршрут повинен тривати не більше трьох днів). Рейси між одними і тими ж аеропортами можуть повторяться кілька разів за розглянутий проміжок часу. Якщо моделювати розклад в якості орієнтованого графа (де аеропорти – це вершини, а рейси – ребра), то отримаємо мультиграф (з кратними ребрами).

Для спрощення реалізації методу гілок і меж граф розклад конвертується в реберний граф. Отриманий граф є також орієнтованим зі зваженими вершинами, але вже не мультиграфом. Ваги для ребер визначаються наступним способом

$$\frac{w(a)+w(b)}{2},$$

щоб перейти назад до графу зі зваженими ребрами, *a,b* – вершини реберного графа, *w(a)* – вага вершини *a*.

2.4 Метод гілок та меж для пошуку цілочисельного розв'язку

Розв'язком початкової задачі вважається отримане необхідне цілочисельне рішення. Якщо найчастіше порушене обмеження виконано, але отримані змінні в основній не цілі, то в алгоритмі Branch and Price запускається метод гілок і меж для пошуку цілочисельного оптимального розв'язку.

За допомогою основної задачі і підзадачі знаходяться оптимальні нецілочисельні значення змінних для початкового завдання. Серед них можуть бути як цілочисельні, так і дробові значення. Розгалуження здійснюється тільки для дрібних змінних з фіксованим значенням цілих. Для кожної змінної реалізується розгалуження по двох гілках: $x_r = 0$ та $x_r = 1$. (див. докладніше Рисунок 2.2)

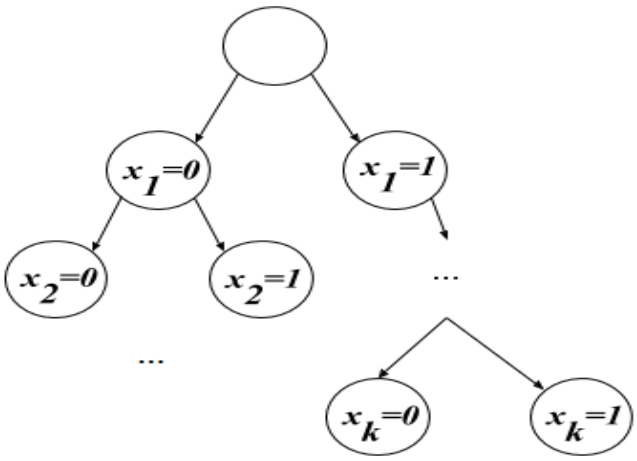


Рисунок 2.2 – Процес розгалуження при пошуку цілочисельного розв'язку

Нижня межа необхідна як і для пошуку найбільш порушуваного обмеження, так і для знаходження цілочисельного розв'язку. Дане значення вже знайдено на попередньому етапі методу. Значення цільової функції вихідної задачі з нецілочисельними змінними є нижньою межею, при порівнянні з якою відсікаються підмножини рішень, які очевидно не є оптимальними.

Існує два сценарії при гілкування $x_r = 0$ та $x_r = 1$, щоб забезпечити реалізацію, описану в даній главі, необхідно повторити всі основні етапи метода при деяких фіксованих змінних. В першому випадку ($x_r = 0$) на основну задачі накладається ще обмеження $x_r \leq 0$. Проте в допоміжній задачі необхідно буде проігнорувати даний маршрут при пошуку початкового рішення. Під час розгалуження для найбільш порушуваного обмеження сумарна вартість забороненого маршруту так само не повинна виступати в якості поточної нижньої границі.

При $x_r = 1$ до основної задачі додається ще обмеження $x_r \geq 1$. У цій гілці немає необхідності ігнорувати даний маршрут, і тому всі етапи відбуваються так само, як і при пошуку нецілочисельного розв'язку.

Розгалуження закінчується, коли всі змінні основної задачі мають значення 0 або 1. Результатом функції є набір маршрутів, який покриває всі обов'язкові рейси і буде виконуватися екіпажами авіакомпанії.

2.5 Адаптований до розробки алгоритм

Для розробки та створення комп'ютерної моделі алгоритм Branch and Price необхідно автоматизувати для уникнення притаманних властивостей для розв'язку даного типу задач. Одна з основних проблем, що виникає при розробці так званої задачі формування графіку команд, – це перебір всієї

множини, що подається на вхід, перш за все через унеможливлення вивести єдину математичну модель. При практичному застосуванні це ключовим чином впливає на ресурси, що необхідні в розробці, такі як час роботи та пам'ять пристрою.

Ще однією структурною особливістю даного типу методів є його неможливість розв'язувати задачі третього та четвертого етапів формування графіку паралельно (маршрутизація та створення розкладу), а це з розвитком сучасних технологій, зокрема багатопоточності, критично збільшує час пошуку рішення задачі.

Тому було вирішено адаптувати наведений вище алгоритм з урахуванням розвитку сучасних мов програмування та з точки зору заощадження ресурсів.

Першим кроком буде подача на вхід списку всіх польотів, серед яких будуть як обов'язкові рейси, що потрібно покрити в будь-якому випадку, та необов'язкові, що не несуть значної економічної вигоди для компанії.

Наступним етапом буде запуск циклу на пошук рейсу, що вилітає найраніше, при цьому жорсткої умови на те, щоб цей рейс був обов'язковим, не накладаємо, що є першою характерною особливістю адаптованого алгоритму. Було вирішено робити цей крок, щоб не пропустити ті рейси, що могли б бути виконані, але через особливості часу вильоту були б позбавлені нашої уваги.

Ще одним цікавим моментом при формуванні даного алгоритму буде залучення до роботи рекурсії. Основним ключем, по якому буде відбуватися перевірка правильності виконання – це час відльоту попереднього та час прильоту наступного.

Критерієм додавання певного рейсу до конкретного циклу будуть витрати, і їх прорахунок на кожному етапі буде необхідним та обов'язковим.

Важливим моментом в уникненні повторів використаних рейсів буде коректне їх видалення після додавання до якого-небудь циклу.

По завершенню виконання програми повинні бути отримані список циклів, кожен з яких міститиме хоча б один обов'язковий рейс, та список з обов'язкових рейсів, що не ввійшли в жоден цикл.

Висновки до Розділу 2

У сучасному світі актуальною проблемою залишається маршрутизації транспортних потоків, під якою розуміємо організовану систему доставки будь-яких матеріальних речей з точки А в точку Б по оптимізованому маршруту. Маршрут розцінюємо замкнений набір рейсів, які бортова команда має змогу за певний проміжок часу.

Важливим було аналіз існуючих алгоритмів з точки зору їх пропонованої ними економічної доцільності, а відтак і спроможності забезпечити сучасні потреби авіакомпаній. Ретельне дослідження даних підходів виявило потребу переосмислення та реорганізації технічної складової, зокрема побудови комп'ютерних моделей на основі алгоритмів дослідженої задачі транспортної маршрутизації.

Особлива увага була звернена на алгоритм Branch and Price, як на одне з останніх досліджень у сфері авіаційних перевезень, зокрема на його структуру та основі складові елементи. Було виявлено, що на етапі основної задачі вкрай доцільно досліджувати окрім максимізації прибутків, ще й умови мінімальних витрат. Врахування цього обмеження було зумовлене введенням додаткового критерію оптимізації, що в подальшому сприятиме удокладненому дослідженню даної проблеми.

Запропонована удосконалена модель розв'язку задачі планування пасажирських авіаперевезень має перевагу в адаптації до наявних в розпорядженні технічних ресурсів, зокрема уникнення переповнення стеку за

рахунок оптимізованого рекурсивного підходу та спроможності введення багатопоточності для функціонально незалежних етапів процесу.

Запропонований алгоритм є універсальним, оскільки здатен мати виходи на інші сфери діяльності (наприклад, може бути застосований для суміжних типів перевезень, окрім тих, маршрут яких не є обмеженим наявністю і протяжністю полотна).

Технологічний прогрес має надзвичайно швидкі темпи, тому ми свідомі того, що подібний алгоритм буде ще не раз удосконалюватися відповідно до нагальних потреб сучасного споживача.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ДЛЯ ПОБУДОВИ ГРАФІКУ ПОЛЬОТІВ СУДЕН ЦИВІЛЬНОЇ АВІАЦІЇ

3.1 Вибір методології розробки програмного продукту

Перш ніж переходити до безпосередньої розробки продукту варто приділити увагу його коректному проектуванню та відповідності останнім світовим практикам у методології життєвого циклу програм. Це є вагомим для економії часу при реалізації та запобіганню стандартних "підводних каменів" у даному типі проектів.

Вибір одного із сучасних підходів до проектування залежить від багатьох факторів: специфікою поставлених завдань розробки, виділених на проект коштів, певних суб'єктивних побажань та ін. Розглянемо чотири основних методології, їх переваги та недоліки, рефлексуючи до сформульованої в попередньому розділі задачі.

– Водоспадна модель (Waterfall Model)

Першою класично розглядають каскадну, або водоспадну, модель циклу розробки програмного забезпечення (див. Рисунок 3.1).

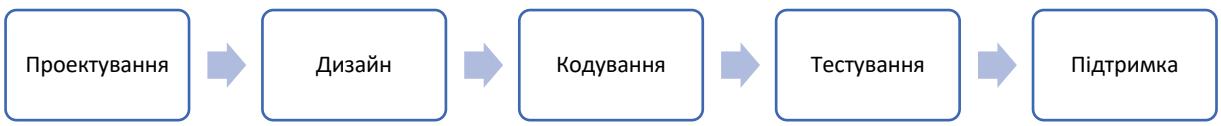


Рисунок 3.1 – Водоспадна модель розробки ПЗ

Така модель є найстарішою і зазвичай використовується тільки в тому випадку, коли:

– вимоги до проекту (технічне завдання) відомі, зрозумілі та чітко сформульовані;
– немає суперечливих завдань;
– у відносно невеликих проектах за умови доступності кваліфікованого персоналу.

Серед основних недоліків такого підходу виділяють складність інтегрування нових компонентів на останніх етапах розробки, коли виявляється більшість помилок, та дороговизна усунення неполадок.

– V-модель

Головною особливістю та відмінністю від попередньої моделі поточної є її полегшене тестування з можливістю внесення коректив на заключних етапах проекту (див. Рисунок 3.2).

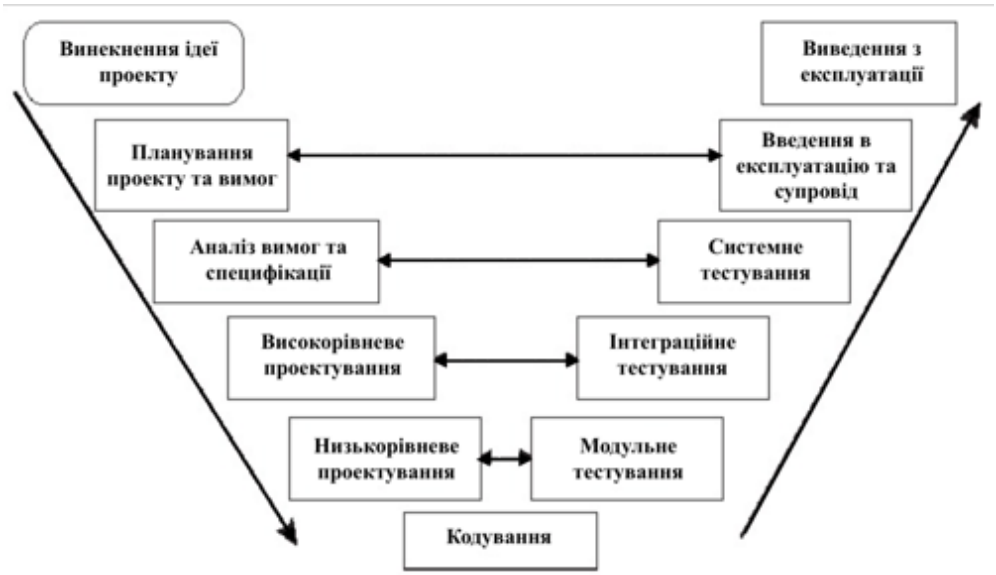


Рисунок 3.2 – V-модель розробки ПЗ

Тому з основних прикладів застосування V-моделі є проекти з достатньою кількістю висококваліфікованого персоналу, зокрема тестувальників, та які потребують численних перевірок.

До недоліків даного підходу відноситься те, що кожна наступна фаза циклу починається виключно по завершенню попередньої, тестування зокрема

здійснюється після того, як продукт буде створено, тому виявлені помилки спричинять "відкидання" на попередні етапи життєвого циклу.

– Інкрементна модель (Incremental Model)

Інкрементні моделі застосовують в тих випадках, коли окремі запити на зміни чітко означені, можуть бути легко формалізовані та піддаються необхідній реалізації (Рисунок 3.3). У той же час деякі деталі проекти можуть дороблюватися в процесі. Також такий життєвий цикл підходить у разі необхідності раннього виводу продукту на ринок або імплементації деяких ризикових характеристик.

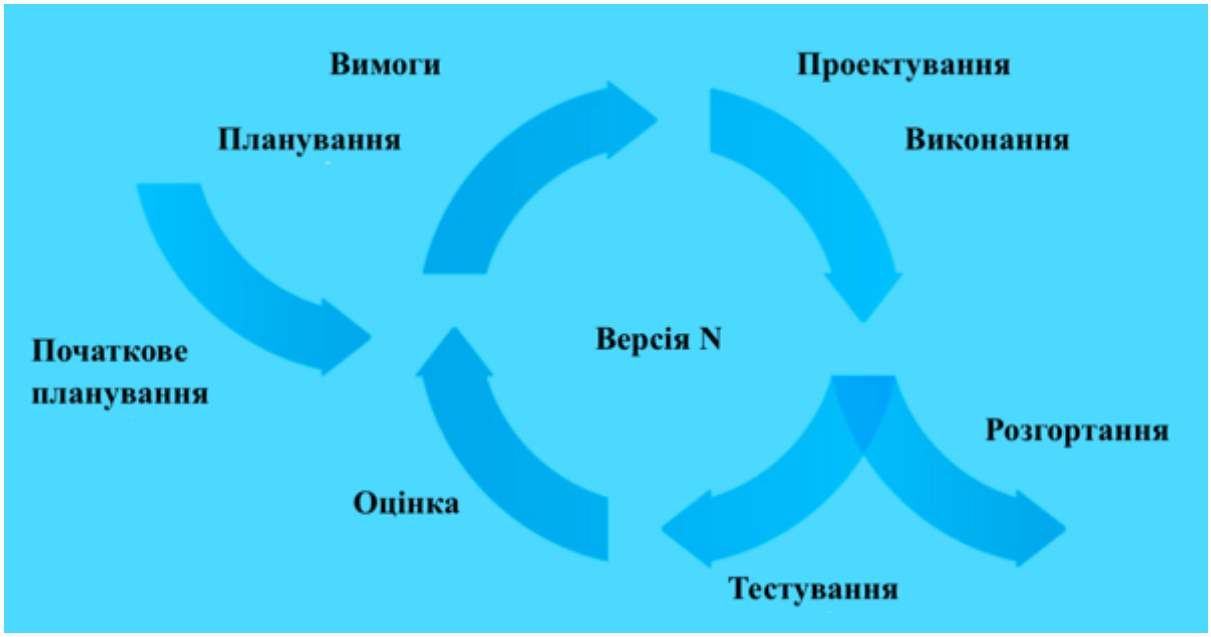


Рисунок 3.3 – Інкрементна модель життєвого циклу

– Гнучка модель (Agile Model)

До основної переваги даного підходу належить той факт, що замовник може спостерігати результат після кожної з ітерацій і корегувати напрямок розвитку у зв'язку з власним вдоволенням проектом (Рисунок 3.4). Головним недоліком "гнучкої" моделі є складність оцінки трудовитрат та вартості розробки через відсутність конкретних формулювань результатів.

До основних випадків, коли при розробці варто застосовувати методологію гнучкого життєвого циклу ПЗ, належать:

– проекти, де потреби користувачів постійно змінюються у зв'язку зі швидкозмінною динамікою бізнесу;
– обставини, коли немає широкого бюджету для реалізації змін і для зменшення собівартості необхідні часті інкременти;
– нагоди на довге та тривале планування немає, і можна застосовувати лише мінімальне планування.

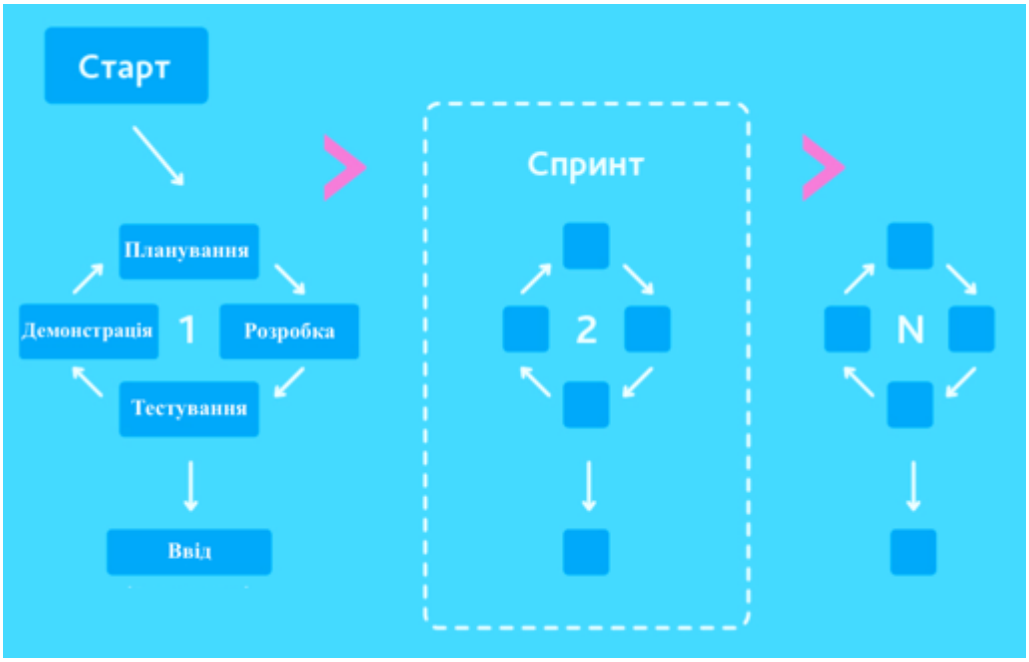


Рисунок 3.4 – "Гнучка" модель розробки

Провівши дослідження присутніх у розпорядженні ресурсів, зокрема наявності чітко сформульованих та описаних вимог до проекту й відсутності будь-яких суперечностей у технічному завданні, а також передбачення потенційно невеликого застосунку, було вирішено зупинити нашу увагу на водоспадній моделі життєвого циклу розробки та дотримуватися її.

3.2 Проектування програмного застосунку та окреслення стеку технологій

Даний етап ознаменовується розв'язанням переліку задач з виключно проектувальної точки зори. Один з найперших – вибір метода та стратегії розв'язку поставленого завдання – був сформульований та окреслений у попередніх розділах. Основою підходу буде служити адаптований до сучасних комп'ютерних потужностей та технологічних реалій алгоритм Branch&Price.

Головною відмінністю розробленого методу є вибір рейсу, який відправляється найраніше, на відміну від обрання найпершого обов'язкового рейсу. Ця особливість була додана у зв'язку з необхідністю побудови максимально повних циклів, які б включали найбільшу можливу кількість рейсів, а не тільки ті, які авіакомпанією вважаються обов'язковими.

Не менш важливим, на нашу думку, також має бути впровадження додаткового критерію оптимізації (витрат), що у свою чергу дасть змогу в майбутньому добудовувати архітектуру над розробленим додатком.

Критично важливим на даному етапі коректно визначитися із загальним стеком технологій, що будуть застосовуватися. Першим на черзі є вибір представлення даних. Для зручності було вирішено використовувати один з найпростіших типів, а саме, цілочисельний int, та втім необхідно передбачити можливість адаптації розробки до реальних запитів, наприклад використання специфічного типу для часу та великих чисел.

За останніми дослідженнями в Україні другою за популярністю є мова розробки Java [24], тому для автоматизації запропонованого алгоритму була застосована Java версії 1.8.

Вибір на користь цього способу розробки був також безпосередньо зумовлений великою кількістю переваг даної мови, зокрема:

- кросплатформенністю та дотриманням терміну "пиши раз - запускай всюди" ("Write once, run anywhere");

– широким застосуванням в різноманітних сучасних індустріальних проектах;
– потужною розвиненою інфраструктурою, що була сформована завдяки десятиліттям використання в програмних розробках.

Система повинна бути призначена для побудови застосувань промислового рівня. Серед сучасного апарату, розробленого для вирішення цієї проблеми, найпопулярнішими є Spring framework та jee, який частково імплементує Spring, та на відміну від нього jee застосовується в переважній більшості для великих проектів з потужностями для подальшої підтримки. Тому зважаючи на це, було вирішено віддати перевагу Spring framework.

Використання Spring framework дає потужні можливості для застосування Spring mvc (model view controller) для генерації веб-сторінок на сервері та Spring dependency injection – технології розробки застосувань, за якої окремі незалежні класи, де чітко відомі, як вони працюють поодинці, зв'язується в одне ціле як застосування.

Основною задачею даного додатку буде реалізація алгоритму, тому було вирішено розробити спрощений варіант сховища, не зв'язаний з базою даних, та який можна за бажанням розширити до роботи з неї, наприклад для зберігання рейсів та циклів або для зчитування інформації.

3.3 Інтерфейс програми та аналіз отриманих результатів

У браузері відкриваємо початкову сторінку додатку (Рисунок 3.5).

Рисунок 3.5 – Початкове вікно програмного продукту

Для найбільш фундаментального ознайомлення з кожним елементом даної форми пропонується розглянути Таблицю 3.1.

Таблиця 3.1 – Огляд початкової форми

Назва елементу	Призначення та особливості
Calculate	Запуск алгоритму та отримання відповіді
Choose file	Обрання файлу (файл .csv з розділювачем "кома")
Upload CSV file	Завантажити обраний файл
Number	Номер рейсу (не може повторюватися)
From	Місце вильоту (не може співпадати з пунктом призначення)
To	Пункт призначення
Income	Прибуток авіакомпанії (максимізується в задачі)
Expanses	Витрати компаній (мінімізуються)
Departure time	Час відправки (не може співпадати з часом прибуття)
Arrival time	Час прибуття
Mandatory	"Обов'язковість" рейсу
Add or update flight	Додати чи оновити рейс
Remove	Видалити рейс (з'являється після додавання хоча б одного рейсу)

Всі дані, окрім поля Mandatory, задаються у спрощеному форматі даних int, а "обов'язковість" рейсу визначається вибором з випадаючого списку true (так) або false (ні).

Програмою передбачена валідація введених даних (Рисунок 3.6). Була розроблена перевірка на різні місця вильоту і прильоту та накладена умова на час відправки для конкретного рейсу (він має бути меншим за час прильоту).

Рисунок 3.6 – Валідація введених даних

Оновлення даних про рейс було вирішено забезпечити за рахунок його номеру. Покроково це виглядатиме наступним чином:

– *крок 1.* Ввели рейс (Рисунок 3.7).

Рисунок 3.7 – Крок 1. Введений рейс

– *крок 2.* Захотіли його змінити. Вводимо нові дані, зберігаючи номер рейсу, який підлягає зміні (Рисунок 3.8).

Рисунок 3.8 – Крок 2. Зміна даних рейсу

– *крок 3.* Отримали оновлений рейс (Рисунок 3.9).

Рисунок 3.9 – Крок 3. Змінений рейс

Розглянемо приклад роботи з даними, що були завантажені з файлу (Рисунок 3.10).

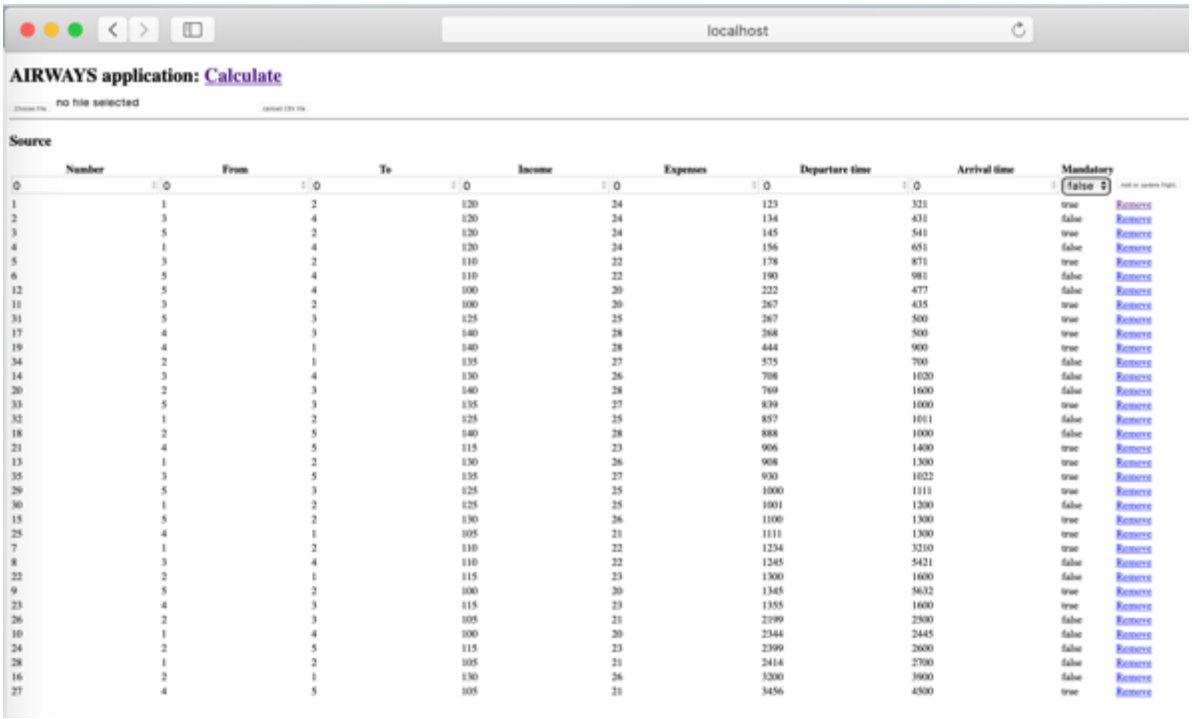


Рисунок 3.10 – Завантажені дані з файлу

Отримали набори циклів, в кожен з яких входить хоча б один з обов'язкових рейсів, з розрахованим часом (Рисунок 3.11).

Detected cycles

from: 4, departure time: 268

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
17	4	3	140	28	268	500	true
14	3	4	130	26	708	1020	false

from: 4, departure time: 444

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
19	4	1	140	28	444	900	true
13	1	2	130	26	908	1300	true
22	2	1	115	23	1300	1600	false
10	1	4	100	20	2344	2445	false

from: 5, departure time: 267

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
31	5	3	125	25	267	500	true
35	3	5	135	27	930	1022	true

from: 2, departure time: 888

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
18	2	5	140	28	888	1000	false
15	5	2	130	26	1100	1300	true

from: 1, departure time: 123

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
1	1	2	120	24	123	321	true
34	2	1	135	27	575	700	false

from: 5, departure time: 145

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
3	5	2	120	24	145	541	true
24	2	5	115	23	2399	2600	false

from: 5, departure time: 222

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
12	5	4	100	20	222	477	false
21	4	5	115	23	906	1400	true

from: 3, departure time: 134

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
2	3	4	120	24	134	431	false
23	4	3	115	23	1355	1600	true

from: 3, departure time: 178

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
5	3	2	110	22	178	871	true
26	2	3	105	21	2199	2500	false

from: 1, departure time: 156

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
4	1	4	120	24	156	651	false
25	4	1	105	21	1111	1300	true

from: 5, departure time: 190

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
6	5	4	110	22	190	981	false
27	4	5	105	21	3456	4500	true

from: 3, departure time: 267

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
11	3	2	100	20	267	435	true
20	2	3	140	28	769	1600	false

Рисунок 3.11 – Результати роботи. Цикли.

Також отримаємо обов'язкові рейси, що не ввійшли в жоден з циклів (Рисунок 3.12).

Detected mandatory flights without cycles

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
33	5	3	135	27	839	1000	true
13	1	2	130	26	908	1300	true
31	5	3	125	25	267	500	true
29	5	3	125	25	1000	1111	true
7	1	2	110	22	1234	3210	true
9	5	2	100	20	1345	5632	true

Рисунок 3.12 – Результати роботи. Обов'язкові рейси без циклів.

Висновки до Розділу 3

Важливим етапом у процесі розробки програмного продукту є його коректне проектування та відповідність новітнім досягненням світової практики у розробці. Вибір підходу зумовлений низкою факторів, зокрема специфікою поставлених завдань розробки, виділених на проект коштів, певних суб'єктивних побажань та ін.

Поширеними моделями циклів розробки програмного забезпечення є водоспадна, V-модель, інкрементна та гнучка.

Водоспадна модель є часто вживаною, має свою специфіку використання, та складність інтегрування нових компонентів на вирішальних етапах розробки.

V-модель вирізняється полегшеним тестуванням на останніх етапах життєвого циклу проекту; вимагає залучення значної кількості тестувальників та витрат часового ресурсу.

Інкрементна модель використовується за умови чіткого окреслення запитів на зміни в задачі; передбачає корекцію задачі на різних етапах її впровадження.

Гнучка модель вирізняється серед інших тим, що замовник бере активну участь в розробці та реалізації проекту на різних його етапах та залишає за собою право внесення коректив. Однак ця модель розробки хибує на реальну оцінки трудових затрат та розрахунку вартості майбутнього проекту.

Враховуючи наявність чітко описаних вимог до проекту й відсутність суперечностей у технічному завданні, а також потенційність невеликого проекту, вважаємо за доцільне закумулювати увагу на каскадній моделі життєвого циклу розробки у досліджуваному підході.

Основою розробленого алгоритму є пристосований до сучасних технологічних потужностей Branch&Price. Врахована специфіка обрання найранішого рейсу на відміну від першого з обов'язкових, що дає можливість сформувати максимально наповнені цикли. Важливо також враховувати можливість розширення архітектури застосунку за рахунок впровадження додаткових критеріїв оптимізації. Актуальним є коректне визначення з технологіями розробки, зокрема з типом даних.

Для реалізації нашого проекту була обрана мова програмування Java та фреймворк застосунків Spring framework, а використання спрощеного варіанту сховища було зумовлено першочерговою необхідність реалізації алгоритму проекту.

Запропонований продукт вирізняється своєю універсальністю та легкою імплементацією новітніх доповнень, що притаманні іншим індустріям, де вирішується задача маршрутизації. Актуалізація уваги виключно на авіаперевезеннях була зумовлена всеохопною локазіцаєю; авіаперевезення акумулюють в собі основні ознаки та моделі впровадження інших видів перевезення.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО - ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ДОДАТКУ ДЛЯ ФОРМУВАННЯ ГРАФІКУ ПОЛЬОТІВ ЛІТАКІВ

4.1 Постановка завдання техніко-економічного дослідження

Головною метою даного розділу є проектування програмного продукту для авіаційних та інших транспортних компаній, що буде сфокусований на оптимізації маршруту судна з можливістю подальшого розширення та доповнення функціоналу. Застосунок розроблений на основі веб-фреймоврка Spring Framework та складається з клієнтської й серверної частин.

4.2 Обґрунтування функцій та параметрів програмного продукту

До основних функцій продукту належать

- F1 - побудова алгоритму розв'язку:
  - А) використання відомого Branch&Price;
  - Б) оновлення відомих підходів у відповідності до потреб замовника;
- F2 - вибір мови розробки:
  - А) Java;
  - Б) Python;
- F3 - обрання інструменту для збирання архіву:
  - А) maven;
  - Б) gradle.

На основі досліджених функцій побудуємо морфологічну карту (Рисунок 4.1).

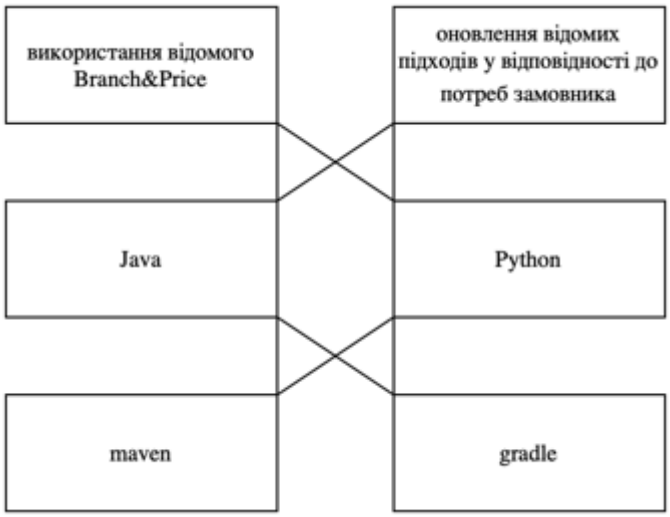


Рисунок 4.1 – Морфологічна карта

Наведемо позитивно-негативну матрицю варіантів основних функцій (див. докладніше Таблицю 4.1).

Таблиця 4.1 – Позитивно-негативна матриця

Основна функція	Варіант реалізації	Переваги	Недоліки
F1	А	Є можливість передбачити результати роботи	Не пристосований до сучасних технічних можливостей
	Б	Змога спроектувати продукт під конкретні вимоги замовника	Потрібно витратити час на узгодження деталей та алгоритму
F2	А	Добре розвинена інфраструктура навколо мови	Необхідний час на поновлення знань щодо застосунків

Продовження Таблиці 4.1 – Позитивно-негативна матриця

	Б	Сильний математичний апарат для впровадження майбутніх побажань клієнта	Складно розбити на модулі для подальшої побудови фреймворків
F3	А	Є можливість простішої імплементації для великих підприємств	Занадто складна архітектура для невеликого проекту
	Б	Застосування простішого синтаксису і отримання легшого файлу як результат	Важко перепрофілювати під великий проект

Для характеристики прототипу програмного продукту використаємо параметри X1 – X3. На основі даних, що наведені в літературі, визначимо мінімальні, середні та максимально допустимі значення (див. Таблицю 4.2).

Таблиця 4.2 – Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Мінім.	*Середнє*	*Максимальне*
Час для огляду теорії (год)	X1	5	15	30
Розробка алгоритму (год)	X2	10	25	45
Час для реалізації (год)	X3	10	30	60

За наведеними в таблиці значеннями параметрів будуємо їх графічне представлення (див. Рисунок 4.2).

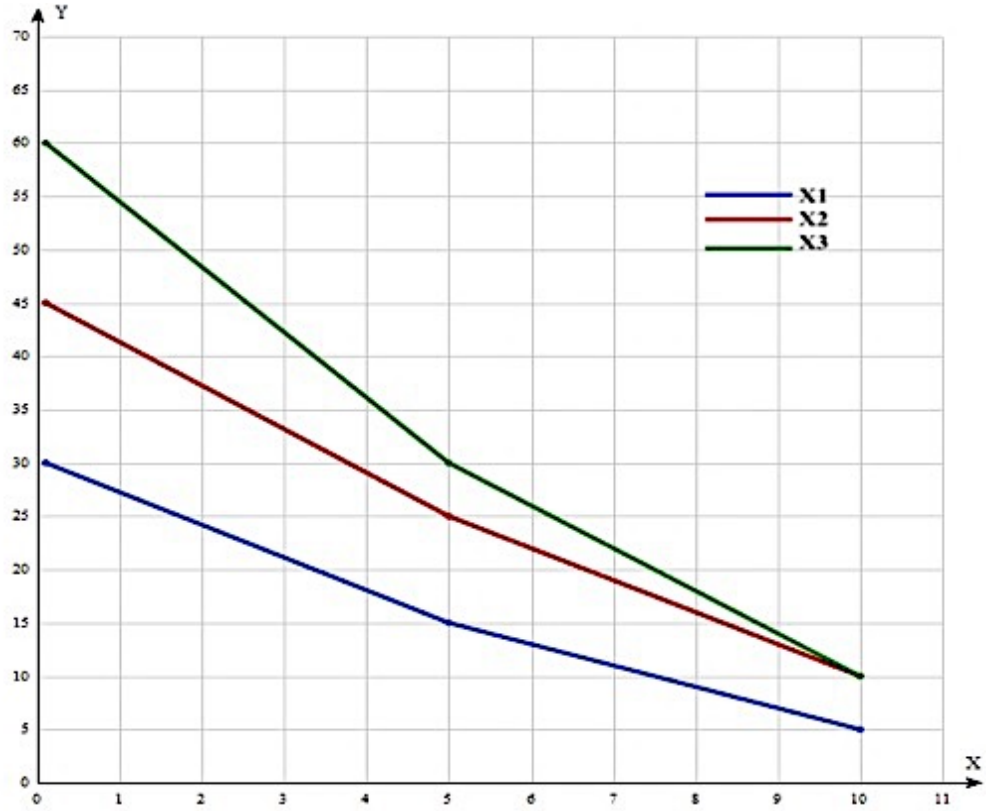


Рисунок 4.2 – Значення параметрів X1, X2, X3

Проведемо оцінку вагомості параметрів методом попарного зрівняння. Ранги варіюються від 1 до 3. За основу найбільшого рангу візьмемо третій, відповідно за найменший буде обрано перший. Результати наведені в Таблицях 4.3-4.4.

Таблиця 4.3 – Результати оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	Відхилення Δ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	*2*	*3*	*4*	*5*	*6*	*7*			
X1	2	2	1	1	1	2	1	10	-4	16
X2	3	3	3	2	3	3	3	20	6	36
X3	1	1	2	3	2	1	2	12	-2	4
Разом	6	6	6	6	6	6	6	42	0	56

Таблиця 4.4 – Попарне зрівняння параметрів

Пара-метр	Експерти							Кінцева оцінка	Числове значення
	1	*2*	*3*	*4*	*5*	*6*	*7*		
X1 та X2	<	<	<	<	<	<	<	<	0.5
X2 та X3	>	>	>	<	>	>	>	>	1.5
X3 та X1	<	<	>	>	>	<	>	>	1.5

Тоді коефіцієнт конкординації буде дорівнювати

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12*56}{36(27-3)} \approx 0.778 > W_k = 0.67$$

Оскільки коефіцієнт конкординації більше нормативного, результати будемо вважати достовірними. Розрахунок вагомості критеріїв наведено в Таблиці 4.5.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри	Параметри x_j			Перший крок		Другий крок	
	X1	*X2*	*X3*	b_i	$K_{вi}$	b_i	$K_{вi}$
X1	1	0.5	0.5	2	0.22	5.5	0.22
X2	1.5	1	1.5	4	0.44	11.5	0.46
X3	0.5	1.5	1	3	0.33	8	0.32
Загалом:				9	1	25	1

4.3 Аналіз варіантів реалізації функцій

Провівши аналіз наведених варіацій, було вирішено залишити наступні:

1. F1(Б) => F2(А) => F3(А);
2. F1(Б) => F2(А) => F3(Б).

Таблиця 4.6 – Обчислення коефіцієнтів рівня якості

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	Б	X1	5	10,00	0,22	1,40
		X2	25	5,00	0,46	2,30
F2	А	X3	30	5,00	0,32	1,80
F3	А	X1	20	3,40	0,22	0,80
		X2	15	8,20	0,46	2,10
	Б	X1	30	0,10	0,22	1,80
		X2	35	2,50	0,46	2,80

Розрахуємо рівні якості для обох варіантів:

$$К1_{F3\ (А)} = 1,40 + 2,30 + 1,80 + 0,80 + 2,10 = 8,40$$

$$К2_{F3\ (Б)} = 1,40 + 2,30 + 1,80 + 1,80 + 2,80 = 10,10$$

Оскільки другий варіант має більший показник рівня якості, то він буде вважатися найкращим.

4.4 Економічний аналіз варіантів розробки

Проведемо розрахунок трудомісткості для визначення вартості розробки. Обидва варіанти реалізації складаються з наступних завдань:

1) Розробка алгоритму на основі вже досліджених підходів;

2) Реалізація програмного продукту з використанням Java.

Кожен з наведених варіантів має додаткове завдання. Для першого варіанту додатковим завданням буде

3.1) напрацювання нової теоретичної бази для правильного застосування типу зборки,

а для другого варіанту

3.2) створення необхідних технічних умов для застосунку.

- *Для завдання 1* - алгоритм групи складності 1, ступінь новизни А, вид використаної інформації – НДІ - трудомісткість складатиме 90 людино-днів, $К_п$= 1,7, $К_{ск}$ =1, $К_{ст}$=0,7. Тому $Т_1$ = 90*1,7*0,7= 107,1 людино-днів.

- *Для завдання 2* - алгоритм групи складності 1, ступінь новизни В, вид використаної інформації – НДІ - трудомісткість буде дорівнювати 43 людино-днів, $К_п$= 0,81, $К_{ск}$ =1, $К_{стм}$= 1. Тоді $Т_2$ = 43*0,81= 34,83 людино-днів.

- *Для додаткового завдання 3.1* - алгоритм групи складності 3, ступінь новизни А, вид використаної інформації НДІ - трудомісткість складає 27

людино-днів, $К_п$ =1,26, $К_{ск}$ = 1, $К_{ст}$=0,7, $К_{стм}$=1. Тому $Т_{3,1}$ = 27*1,26 = 34,02 людино-днів.

– *Для додаткового завдання 3.2* - алгоритм групи складності 3, ступінь новизни В, вид використаної інформації НДІ - трудомісткість - 12 людино-днів, $К_п$ =0.6, $К_{ск}$ = 1, $К_{ст}$ = 0.7, $К_{стм}$=1. Отримаємо, що $Т_{3,2}$ = 12*0,6 = 7,2 людино-днів.

Отримаємо такі результати трудомісткості для обох варіантів

$$Т_1 = 34,83 + 107,1 + 34,02 = 175,95$$
$$Т_2 = 34,83 + 107,1 + 7,2 = 149,13$$

При розробці програмного продукту варто врахувати залучення програміста із зарплатою 20 000 грн та аналітика, який отримуватиме 15 000 грн. Тоді погодинна заробітна плата працівників складатиме

$$С = (20\,000 + 15\,000) / 2 * 21 * 8 = 104,16 \text{ грн}$$

Для кожного з варіантів

– $С_1$= 175,95 * 104,16 * 1,2 * 8 = 175 938,74 грн
– $С_2$ =149,13 * 104,16 * 1,2 * 8 = 149 120,46 грн

Відрахування на ЄСВ складає 22%, або для кожного з варіантів

– $С_1'$ =$С_1$ * 0,22 = 175 938,74 * 0,22 = 38 706,52 грн
– $С_2'$ =$С_2$ * 0,22 = 149 120,46 * 0,22 = 32 806,50 грн

Наступним кроком визначмо витрати на оплату однієї машино-години. З урахуванням зарплати програміста розміром в 20 000 грн та з коефіцієнтом зайнятості 0,2 отримаємо

$$С_г = 12 * 20\,000 * 0,2 = 48\,000 \text{ грн}$$

Урахувавши додаткову заробітну плату

$$С_{зп} = С_г * (1 + 0,2) = 48\,000 * 1,2 = 57\,600 \text{ грн}$$

Відрахування на соціальний внесок складають 22%

$$С_{від} = С_{зп} * 0,22 = 57\,600 * 0,22 = 12\,672 \text{ грн}$$

Амортизаційні відрахування розраховуємо за умови, що амортизація дорівнює 25%, а вартість ЕОМ становить 15 000 грн:

$$С_А = К_{ТМ} * К_А * Ц_{ПР} = 1,15 * 0,25 * 15\,000 = 4\,312,50 \text{ грн}$$

Розрахуємо витрати на ремонт та профілактику:

$$С_Р = К_{ТМ} * Ц_{ПР} * К_Р = 1,15 * 15\,000 * 0,05 = 862,50 \text{ грн}$$

Розрахуємо ефективний годинний фонд часу ПК за рік (в годинах)

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) * t_З * К_В = (365 – 104 – 11 – 16) * 8 * 0,9 = 1\,684,80$$

Тоді отримаємо, що витрати на оплату електроенергії, з урахуванням ПДВ, складатимуть

$$C_{ЕЛ} = 1\,684{,}8 * 0{,}2 * 0{,}2 * 1{,}75 = 117{,}94 \text{ грн}$$

Накладні витрати складуть

$$С_н = Ц_{пр} * 0{,}67 = 15\,000 * 0{,}67 = 10\,050 \text{ грн}$$

Тоді експлуатаційні витрати (грн) дорівнюватимуть

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н = 57\,600 + 12\,672 + 4\,312{,}50 + 862{,}50 + +117{,}94 + 10\,050 = 85\,614{,}94$ грн

А собівартість однієї машино-години дорівнюватиме

$$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 85\,614{,}94 / 1684{,}80 = 50{,}82 \text{ грн/год}$$

Врахувавши, що всі роботи проводяться на ЕОМ, то витрати на оплату машинного часу становитимуть

- $С_{М1} = 50{,}82 * 175{,}95 = 8\,941{,}78$ грн
- $С_{М2} = 50{,}82 * 149{,}13 = 7\,578{,}77$ грн

Накладні витрати становлять 67% від заробітної плати тобто

- $С_{Н1} = 0.67 * 175\,938{,}74 = 117\,878{,}96$ грн
- $С_{Н2} = 0.67 * 149\,120{,}46 = 99\,910{,}70$ грн

Таким чином вартість розробки програмного продукту та проведення дослідів розраховуватимемо за формулою

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

і дорівнюватиме:

- $С_{ПП1}$ = 175 938,74 + 38 706,52 + 8 941,78 + 117 878,96 = 341 466 грн
- $С_{ПП2}$ = 149 120,46 + 32 806,50 + 7 578,77 + 99 910,70 = 289 416,43 грн

4.5 Вибір кращого варіанту програмного продукту техніко-економічного рівня

Коефіцієнти техніко-економічного рівня розраховуються за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}$$

- $К_{тер1}$ = 8,40 / 341 466 = 2,45 *10^{-5}
- $К_{тер2}$ = 10,10 / 289 416,43 = 3,48 *10^{-5}

Висновки до Розділу 4

Докладний аналіз підходів до функціонально-вартісного аналізу програмного додатку дав змогу закумулювати увагу на другому підході розробки застосунку завдяки обчисленому коефіцієнту техніко-економічного рівня, що складає 3,48 *10^{-5}.

ВИСНОВКИ

Сучасна авіація має довгу історію становлення та розвитку: від перших польоту братів Райт до популяризації реактивних двигунів минуло вже більше 110 років.

Перша та Друга світові війни дали поштовх як у сфері технічного прогресу, так і в підготовці висококваліфікованого персоналу. Стрімкий розвиток авіації сприяв здійсненню трансконтинентальних повітряних перевезень, що спонукало до розвитку світової торгівлі та міжкультурній взаємодії.

Авіаперевезення пасажирів та вантажу за багатьма показниками є найефективнішим та найбільш оптимальним способом пересування.

Цивільна авіація ні на секунду не припиняє свого розвитку. Насамперед це відчутно в питаннях безпеки та боротьбі з терористичними загрозами сучасності. Виклики, пов'язані із захистом навколишнього середовища, теж відіграють не останню роль в планах на майбутнє авіаіндустрії.

Розвиток сучасної авіації досяг нечуваного донині піку, оскільки повітряні перевезення не заангажовані локаційно. Основні обмеження, що корелюють роботу повітряного транспорту та його взаємодію з транзитними країнами, були створені людиною, зокрема Свободи повітряного транспорту, затверджені Чиказькою конвенцією (1944 р.).

Невід'ємною складовою перевезень в авіаіндустрії стала поява окремих вантажних служб авіатранспорту. Проте існують оператори, які задовольняють як потреби мандрівників, так і власників вантажу.

Сезонність в авіаперевезеннях теж відіграє значну роль в плануванні графіку польотів, зокрема піком туристичного сезону в Європі та Північній Америці є липень та серпень, а для вантажних перевезень часова структура зміщується на грудень та січень.

Значну роль в організації сучасної авіаційної індустрії відіграють домашні аеропорти найбільших перевізників. Саме в розробці коректного графіку польотів між цими хабами і полягає провідне завдання спеціалістів з транспортної маршрутизації.

Повітряні альянси, членами яких є 50+ найбільших авіакомпаній, задовольняють потреби своїх клієнтів навіть на тих ринках, де окрема компанія слабо представлена або не оперує взагалі.

У коректній взаємодії 11 тисяч пасажирських літаків, що щомиті перебувають в небі, чималу роль відіграє правильно спланований графік польотів, процес якого складається з чотирьох основних етапів (створення розкладу польотів; призначення борту; маршрутизації; створення графіку для роботи вже відповідного екіпажу).

Дана робота розглядає саме останній етап створення розкладу, який моделюється як детермінована задача лінійного програмування, за мету якого покладено мінімізацію витрат авіакомпанії на зарплати членів екіпажу літака.

Хоча дана задача тісно пов'язана із питанням транспортної маршрутизації, лише останні 50 років їй приділяється належна увага. Перші розробки та запропоновані методи були в більшості своїй обмежені потужностями тодішніх електронно-обчислювальних машин, але ця перепона з часом була подолана і вже у 2004 р. був запропонований метод Branch and Price, удосконалення якого у відповідності з потужностями сучасної техніки і стало однією з основних цілей дослідження.

Досі лишається актуальною проблема маршрутизації транспортних потоків, тому аналіз існуючих алгоритмів з точки зору їх економічної доцільності, а відтак і спроможності забезпечити сучасні потреби авіакомпаній є доцільним і вкрай потрібним. Ретельне дослідження даних підходів виявило потребу переосмислення та реорганізації технічної складової, зокрема побудови комп'ютерних моделей на основі алгоритмів дослідженої задачі транспортної маршрутизації.

Важливим лишається алгоритм Branch and Price, як одне з останніх досліджень у сфері авіаційних перевезень. Було виявлено, що на етапі основної задачі вкрай доцільно досліджувати окрім максимізації прибутків, ще й умови мінімальних витрат. Врахування цього обмеження було зумовлене введенням додаткового критерію оптимізації, що в подальшому сприятиме удокладненому дослідженню даної проблеми.

Запропонована удосконалена модель розв'язку задачі планування пасажирських авіаперевезень має перевагу в адаптації до наявних в розпорядженні технічних ресурсів.

Описаний алгоритм є універсальним, оскільки здатен мати виходи на забезпечення інших сфер людської життєдіяльності.

Технологічний прогрес має надзвичайно швидкі темпи, тому ми свідомі того, що подібний алгоритм буде ще не раз удосконалюватися відповідно до нагальних потреб сучасного споживача.

Важливим етапом у процесі розробки програмного продукту є його коректне проектування та відповідність новітнім досягненням світової практики у розробці. Вибір підходу зумовлений низкою факторів, зокрема специфікою поставлених завдань розробки, виділених на проект коштів, певних суб'єктивних побажань та ін.

Поширеними моделями циклів розробки програмного забезпечення є водоспадна, V-модель, інкрементна та гнучка.

Водоспадна модель є часто вживаною, має свою специфіку використання, та складність інтегрування нових компонентів на вирішальних етапах розробки.

V-модель вирізняється полегшеним тестуванням на останніх етапах життєвого циклу проекту; вимагає залучення значної кількості тестувальників та витрат часового ресурсу.

Інкрементна модель використовується за умови чіткого окреслення запитів на зміни в задачі; передбачає корекцію задачі на різних етапах її впровадження.

Гнучка модель вирізняється серед інших тим, що замовник бере активну участь в розробці та реалізації проекту на різних його етапах та залишає за собою право внесення коректив. Однак ця модель розробки хибує на реальну оцінки трудових затрат та розрахунку вартості майбутнього проекту.

Враховуючи наявність чітко описаних вимог до проекту й відсутність суперечностей у технічному завданні, а також потенційність невеликого проекту, вважаємо за доцільне закумулювати увагу на каскадній моделі життєвого циклу розробки у досліджуваному підході.

Основою розробленого алгоритму є Branch&Price. Взята до уваги специфіка обрання найбільш раннього рейсу на відміну від першого з обов'язкових, що дає можливість сформувати максимально наповнені цикли. Була врахована також можливість розширення архітектури застосунку за рахунок впровадження додаткових критеріїв оптимізації.

Обрана мова програмування Java та фреймворк застосунків Spring framework дала можливість реалізувати проект.

Запропонований продукт вирізняється своєю універсальністю та легкою імплементацією новітніх доповнень. Авіаперевезення акумулюють в собі основні ознаки та моделі впровадження інших видів перевезення, закрема закгуалізування уваги виключно на авіаперевезеннях було зумовлене всеохопною територіальною локазіцаєю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Number of flights performed by the global airline industry from 2004 to 2020. URL: https://www.statista.com/statistics/564769/airline-industry-number-of-flights/ (Last accessed: 17.03.2020).
2. Хто користувався українською небом в 2019 році: ТОП-10 авіакомпаній. URL: https://biz.liga.net/all/transport/novosti/kto-chasche-vseh-letal-nad-ukrainoy-v-2019-godu-infografika (дата звернення: 18.03.2020).
3. Employment. URL: https://aviationbenefits.org/economic-growth/supporting-employment// (Last accessed: 19.03.2020).
4. Which major expenses affect airline companies? URL: https://www.investopedia.com/ask/answers/040715/what-are-major-expenses-affect-companies-airline-industry.asp (Last accessed: 20.03.2020).
5. Air Transport. URL: https://transportgeography.org/?page_id=1765 (Last accessed: 21.03.2020).
6. Кількість літаків в небі побило рекорд. URL: https://lenta.ru/news/2018/07/02/recordflightnumber/ (дата звернення: 22.03.2020).
7. Barnhart, C., Cohn, A.M., Johnson,E.L, Klabjan,D., Nemhauser, G.L., Vance,P.H. *Handbook of Transportation Science*, Springer Link, 2003. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.5219&rep=rep1&type=pdf (Last accessed: 22.03.2020).
8. Andersson, E., Housos, E., Kohl, N., Wedelin D. *Operations Research in the Airline Industry*, pp.228-258, 1998. URL: http://www.cse.chalmers.se/~dag/paper/crew.pdf (Last accessed: 22.03.2020).
9. Andres, V., Valdes V. Integrating Crew Scheduling and Rostering Problems. 2010. URL: http://amsdottorato.unibo.it/2705/1/vera_valdes_victor_andres_tesi.pdf (Last accessed: 23.03.2020).
10. Smallen.D. 2015 U.S.-Based Airline Traffic Data. *Washington: Bureau of Transportation Statistics*, 2016.- URL: http://www.rita.dot.gov/bts/press_releases/bts018_16Archetti, (Last accessed: 23.03.2020).

11. C., Speranza M.G. A survey on matheuristics for routing problems. 2013. URL: https://www.unibs.it/sites/default/files/ricerca/allegati/quaderno%20wpdem%2011.pdf (Last accessed: 23.03.2020).
12. Özdemir U. Methodology for crew-pairing problem in airline crew scheduling. 2004. URL: https://www.cmpe.boun.edu.tr/~gungort/theses/Methodology%20for%20Crew-Pairing%20Problem%20in%20Airline%20Crew%20Scheduling.pdf (Last accessed: 23.03.2020).
13. Wren A. A review of computer scheduling of buses and crews. Proceedings of Public Transport Analysis Seminar, pages 42–47, 1968
14. Nicoletti B. Automatic crew rostering. *Transportation Science*, 9(1):33–42, 1975. URL: http://www.academia.edu/4119857/Automatic_Crew_Rostering (Last accessed: 24.03.2020).
15. Freling, R., Lentink, R., Wagelmans A. A decision support system for crew planning in passenger transportation using a flexible branchand-price algorithm. *Annals of Operations Research: Staff Scheduling and Rostering: Theory and Applications*, Part I, 127(1–4):203–222, 2004. URL: https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&m1r_RyWg&sig2=kvC353p4ZvXHlBZOHVBqIg (Last accessed: 25.03.2020).
16. Branch and price. URL: https://en.wikipedia.org/wiki/Branch_and_price (Last accessed: 26.03.2020).
17. Barnhart.C., Marla.L., Jiang,H. Optimization approaches to airline industry challenges: Airline Schedule Planning and Recovery. *DROPS*, 2009. URL: http://drops.dagstuhl.de/opus/volltexte/2009/2188/pdf/09261.BarnhartCynthia.ExtAbstract.2188.pdf (Last accessed: 27.03.2020).
18. Vance, P. H., Atamturk, A. Technical Report TLI/LEC-97- 06, Georgia Institute of Technology, Atlanta, GA, 1997. URL: http://www.ieor.berkeley.edu/~atamturk/pubs/united.pdf (Last accessed: 27.03.2020).

19. Doig, A. G., Land A.H. An automatic method of solving discrete programming problems. *Econometrica*. 28 (3). pp. 497–520, 1960. URL: http://www.jstor.org/stable/1910129?origin=crossref (Last accessed: 23.03.2020).
20. Ehrgott, M., Tind J. Column Generation in Integer Programming with Applications in Multicriteria. *Optimization. Technical Report of the Faculty of Engineering*, 2007. URL: http://www.math.ku.dk/~tind/integercolumnWEB (Last accessed: 22.03.2020).
21. Savelsbergh M. A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Georgia Tech*, 1997. URL: http://www2.isye.gatech.edu/~ms79/publications/or45.pdf (Last accessed: 24.03.2020).
22. Vanderbeck, F., Wolsey, L. An exact algorithm for IP column generation. *Operations Research Letters*, 19, 151 – 159, 1995. URL: https://www.math.u-bordeaux.fr/~fvanderb/papers/ipcgPap.pdf (Last accessed: 26.03.2020).
23. Рейтинг мов програмування 2020. URL: https://dou.ua/lenta/articles/language-rating-jan-2020/ (Last accessed: 28.03.2020).

ДОДАТОК А ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДОПОВІДІ

ДИПЛОМНА РОБОТА НА ТЕМУ:
«СИСТЕМА ПЛАНУВАННЯ ТА ОПТИМІЗАЦІЇ ГРАФІКУ ПОЛЬОТУ ЕКІПАЖІВ НА ОСНОВІ МЕТОДУ BRANCH AND PRICE»

ВИКОНАВИЦЯ РОБОТИ:
СТУДЕНТКА IV КУРСУ,
ГРУПИ КА-63
ПАВЛОЩУК ОЛЬГА ОЛЕКСАНДРІВНА

КЕРІВНИК:
КАНДИДАТ ТЕХ. НАУК, ДОЦЕНТ
ТИМОЩУК ОКСАНА ЛЕОНІДІВНА

Київ-2020

- **Актуальність дослідження** зумовлена необхідністю пізнання еволюції становлення сучасної авіації, механізмів її діяльності та створенням алгоритмів удосконалення ефективності функціонування відповідно до нагальних потреб сьогодення.

- **Мета роботи** – дослідження історії та окреслення еволюції існуючих методів транспортної маршрутизації; визначення чинників динаміки алгоритму Branch and Price; створення комп'ютерної моделі запропонованих змін в удосконаленому алгоритмі транспортної маршрутизації.

- **Об'єкт дослідження** – алгоритм оптимізації графіку польоту екіпажів на основі методу Branch and Price.

- **Методи дослідження –** специфіка об'єкту і поставлені завдання зумовили застосування загальнонаукових (аналіз, синтакс) і спеціальних методів і прийомів (описовий, системно-класифікаційний, функційного аналізу, прийом кількісних підрахунків, прийом трансформаційного аналізу).

- **Наукова новизна** полягає в тому, що вперше досліджено алгоритм Branch and Price як системний динамічний феномен, розроблено і застосовано для його пізнання комплекс прийомів, окреслено механізми, джерела та чинники розвитку сучасних авіаперевезень.

- **Результатом роботи** є розроблений програмний продукт для побудови графіку польотів суден цивільної авіації.

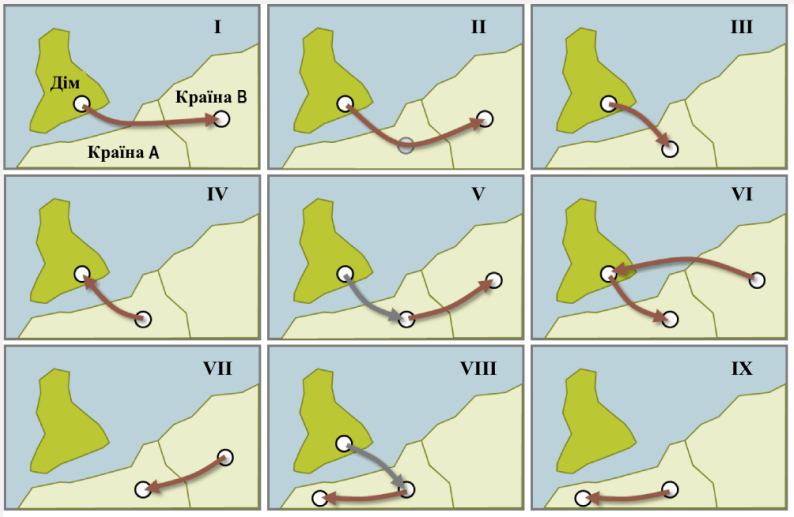
Топ-10 авіаперевізників України (2019 р.)

Міжнародні Авіалінії України	*[58,772 (-4.5%)]*
Turkish Airlines	*[33,716 (+12.5%)]*
WizzAir	*[20,944 (+37.3%)]*
Belavia	*[18,629 (+16.4%)]*
LOT Polish airlines	*[18,449 (+16.7%))]*
SkyUp	*[10,631 (+340.9%)]*
Windrose	*[10,185 (+9.5%)]*
RyanAir	*[9,295 (+437.9%)]*
AzurAir	*[7,229 (+48.8%)]*
Qatar Airways	*[5,893 (+26.1%)]*

Зайнятість в авіаіндустрії

- Оператори аеропорту *(525,000)*
- Інші ролі аеропорту *(оренда авто, прикордонники та ін. - 5.6 мільйонів)*
- Авіалінії *(команди літака, посадки та ін. - 2.7 мільйонів)*
- Цивільний авіапростір *(інженери та дизайнери цивільних літаків та ін. - 1.2 мільйонів)*
- Провайдери авіанавігаційних послуг *(233,000)*

Freedoms of the air (1944 p.)



Формування графіку польотів

Постановка задачі

$$\sum_{r \in R} c_r x_r \to min$$

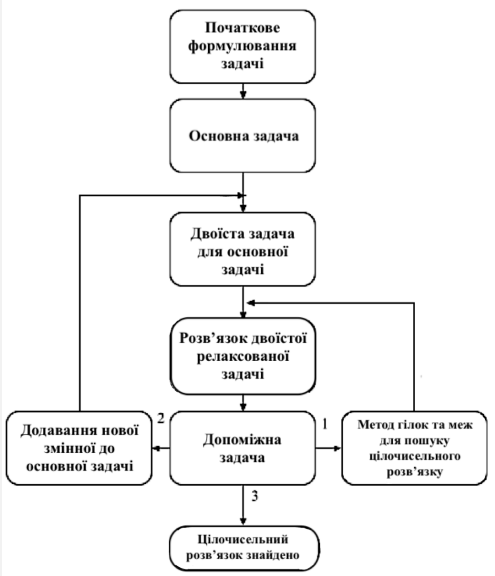
$$\sum_{r \in R} a_{ir} x_r = 1, \forall i \in I^*$$

$$a_{ir} = \begin{cases} 1, & \text{якщо } i \in r \\ 0, & \text{в іншому випадку} \end{cases}$$

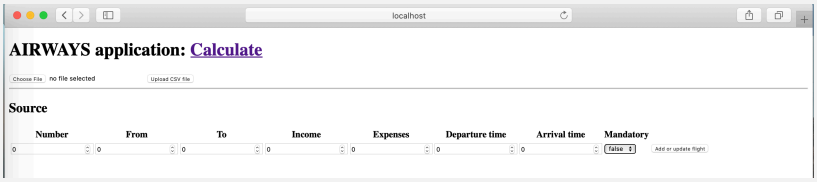
$$x_r = \begin{cases} 1, & \text{якщо } r \text{ входить в розв'язок} \\ 0, & \text{в іншому випадку} \end{cases}$$

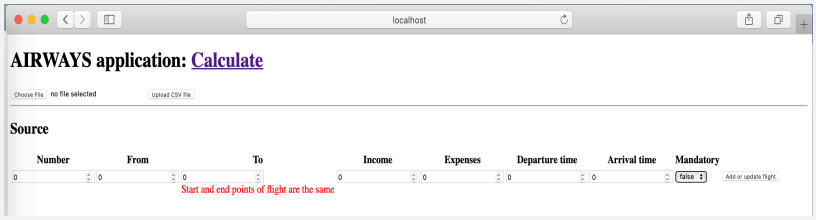
$c_r = \sum_{r \in R} c_i$ – вартість маршруту r

Метод Branch and Price

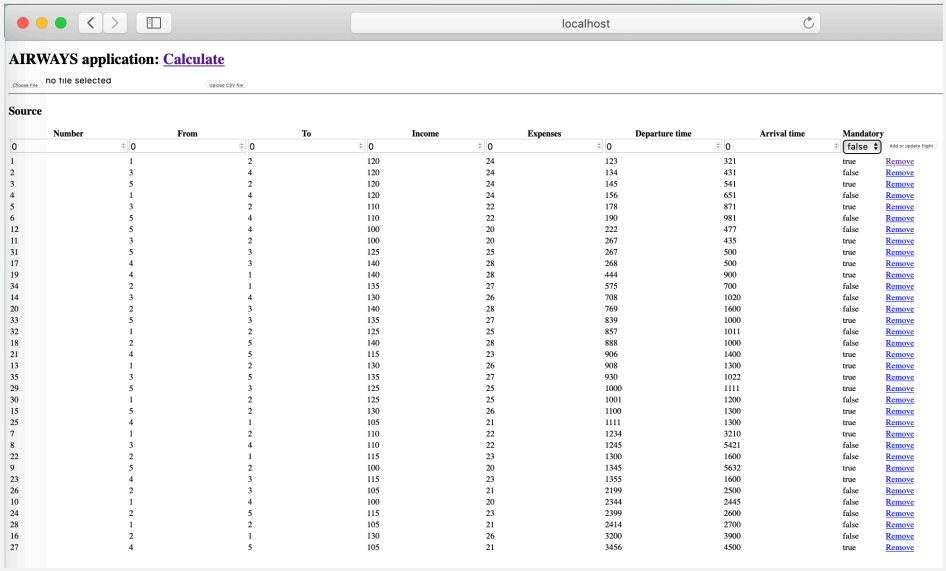


Результати роботи





Результати роботи



	Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory	
0		0	0	0	0	0	0	false	Add or update Flight...
1		1	2	120	24	123	321	true	Remove
2		3	4	120	24	134	431	false	Remove
3		5	2	120	24	145	541	true	Remove
4		1	4	120	24	156	651	false	Remove
5		3	2	110	22	178	871	true	Remove
6		5	4	110	22	190	981	false	Remove
12		5	4	100	20	222	477	false	Remove
11		3	2	100	20	267	435	true	Remove
31		5	3	125	25	267	500	true	Remove
17		4	3	140	28	268	500	true	Remove
19		4	1	140	28	444	900	true	Remove
34		2	1	135	27	575	700	false	Remove
14		3	4	130	26	708	1020	false	Remove
20		2	3	140	28	769	1600	false	Remove
33		5	3	135	27	839	1000	true	Remove
32		1	2	125	25	857	1011	false	Remove
18		2	5	140	28	888	1000	false	Remove
21		4	5	115	23	906	1400	true	Remove
13		1	2	130	26	908	1300	true	Remove
35		3	5	135	27	930	1022	true	Remove
29		5	3	125	25	1000	1111	true	Remove
30		1	2	125	25	1001	1200	false	Remove
15		5	2	130	26	1100	1300	true	Remove
25		4	1	105	21	1111	1300	true	Remove
7		1	2	110	22	1234	3210	true	Remove
8		3	4	110	22	1245	5421	false	Remove
22		2	1	115	23	1300	1600	false	Remove
9		5	2	100	20	1345	5632	true	Remove
23		4	3	115	23	1355	1600	true	Remove
26		2	3	105	21	2199	2500	false	Remove
10		1	4	100	20	2344	2445	false	Remove
24		2	5	115	23	2399	2600	false	Remove
28		1	2	105	21	2414	2700	false	Remove
16		2	1	130	26	3200	3900	false	Remove
27		4	5	105	21	3456	4500	true	Remove

Detected cycles

from: 4, departure time: 268

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
17	4	3	140	28	268	500	true
14	3	4	130	26	708	1020	false

from: 4, departure time: 444

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
19	4	1	140	28	444	900	true
13	1	2	130	26	908	1300	true
22	2	1	115	23	1300	1600	false
10	1	4	100	20	2344	2445	false

from: 5, departure time: 267

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
31	5	3	125	25	267	500	true
35	3	5	135	27	930	1022	true

from: 2, departure time: 888

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
18	2	5	140	28	888	1000	false
15	5	2	130	26	1100	1300	true

from: 1, departure time: 123

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
1	1	2	120	24	123	321	true
34	2	1	135	27	575	700	false

from: 5, departure time: 145

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
3	5	2	120	24	145	541	true
24	2	5	115	23	2399	2600	false

from: 5, departure time: 222

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
12	5	4	100	20	222	477	false
21	4	5	115	23	906	1400	true

from: 3, departure time: 134

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
2	3	4	120	24	134	431	false
23	4	3	115	23	1355	1600	true

from: 3, departure time: 178

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
5	3	2	110	22	178	871	true
26	2	3	105	21	2199	2500	false

from: 1, departure time: 156

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
4	1	4	120	24	156	651	false
25	4	1	105	21	1111	1300	true

from: 5, departure time: 190

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
6	5	4	110	22	190	981	false
27	4	5	105	21	3456	4500	true

from: 3, departure time: 267

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
11	3	2	100	20	267	435	true
20	2	3	140	28	769	1600	false

Detected mandatory flights without cycles

Number	From	To	Income	Expenses	Departure time	Arrival time	Mandatory
33	5	3	135	27	839	1000	true
13	1	2	130	26	908	1300	true
31	5	3	125	25	267	500	true
29	5	3	125	25	1000	1111	true
7	1	2	110	22	1234	3210	true
9	5	2	100	20	1345	5632	true

ВИСНОВКИ

- розроблений програмний продукт для побудови графіку польотів суден цивільної авіації;
- досліджено історію та окреслено еволюцію існуючих методів транспортної маршрутизації
- визначено чинники динаміки алгоритму Branch and Price
- вперше досліджено алгоритм Branch and Price як системний динамічний феномен, розроблено і застосовано для його пізнання комплекс прийомів, окреслено механізми, джерела та чинники розвитку сучасних авіаперевезень;

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б ТЕКСТ ПРОГРАМИ

```
package edu.kpi.ipsa.opavloshchuk.airways.calculation;
import edu.kpi.ipsa.opavloshchuk.airways.data.Cycle;
import edu.kpi.ipsa.opavloshchuk.airways.data.Flight;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Optional;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public class Calculator {
    private final List<Flight> allFlights = new ArrayList<>(); // всі рейси
    private final List<Cycle> cycles = new ArrayList<>(); // обчислені цикли
    private final List<Flight> mandatoryFlights = new ArrayList<>(); // обов'язкові
рейси
    private final List<Flight> mandatoryFlightsWithoutCycles = new
ArrayList<>(); // обчислені обов'язкові рейси поза циклами
    public Calculator(List<Flight> allFlights) {
        if (allFlights == null) {
            throw new IllegalArgumentException("all flights list is null");
        }
        // Робляться копії списків для можливості видалення елементів в
процесі роботи алгоритму
        this.allFlights.addAll(allFlights);
this.allFlights.stream().filter(Flight::isMandatory).forEach(mandatoryFlights::add);
    }
    public List<Cycle> getCycles() {
```

```
        return cycles;
    }
    public List<Flight> getMandatoryFlightsWithoutCycles() {
        return mandatoryFlightsWithoutCycles;
    }
    public void perform() {
        List<Cycle> allCycles = new ArrayList<>();
        // Для всіх рейсів
        while (!allFlights.isEmpty()) {
            // Порядок перебору - від рейсів, що починають найраніше
            final Flight origin = getEarliestFlight();
            // Знайти цикли, що містять обов'язкові рейси
            getCyclesWithMandatoryFlights(origin).forEach(allCycles::add);
            // Видалити рейс, для якого ми шукали цикли - він вже не потрібний
            allFlights.remove(origin);
        }
        // Для всіх обов'язкових рейсів
        while (!mandatoryFlights.isEmpty()) {
            // Порядок перебору - від рейсів із найбільшим прибутком
            final Flight mostValuable = getMandatoryFlightWithMaxIncome();
            // Знайти для такого рейсу найдешевший за витратами цикл
            final Optional<Cycle> cheapestOpt = getCheepestCycle(allCycles,
mostValuable);
            // Видалити рейс, для якого ми шукали цикли - він вже не потрібний
            mandatoryFlights.remove(mostValuable);
            if (cheapestOpt.isPresent()) {
                final Cycle cheapest = cheapestOpt.get();
                // Добавити знайдений найдешевший цикл до результату
                cycles.add(cheapest);
                // Видалити всі цикли, які мають рейси, спільні зі знайденим - рейс
```

```
відбувається лише раз
                allCycles = removeIntersectedCycles(allCycles, cheapest);
            } else {
                // Жодного циклу із таким рейсом не знайдено
                mandatoryFlightsWithoutCycles.add(mostValuable);
            }
        }
    }
    private Optional<Cycle> getCheepestCycle(List<Cycle> allCycles, Flight
flight) {
        return allCycles.stream()
                .filter(cycle -> cycle.contains(flight))
                .min((c1, c2) -> c1.getExpenses(Calculator::getWaitExpenses) -
c2.getExpenses(Calculator::getWaitExpenses));
    }
    private List<Cycle> removeIntersectedCycles(List<Cycle> list, Cycle
baseCycle) {
        return list.stream()
                .filter(cycle -> cycle.getFlights().stream().noneMatch(flight ->
baseCycle.contains(flight)))
                .collect(Collectors.toList());
    }
    private Stream<Cycle> getCyclesWithMandatoryFlights(Flight origin) {
        return detectCycles(new
Cycle(origin)).stream().filter(Cycle::containsMandatory);
    }

        private Flight getEarliestFlight() {
        return allFlights.stream().min((f1, f2) -> f1.getDepartureTime() -
f2.getDepartureTime())
```

```
            .orElseThrow(() -> new NoSuchElementException());
    }
    private Flight getMandatoryFlightWithMaxIncome() {
        return mandatoryFlights.stream().max((f1, f2) -> f1.getIncome() -
f2.getIncome())
            .orElseThrow(() -> new NoSuchElementException());
    }

    private List<Cycle> detectCycles(Cycle base) {
        final Flight last = base.getLast();
        // Перевірити лише цикли, що мають хоча два рейси, отже, можуть
бути замкненими
        if (base.containsBeforeLast()) {
            // Час відправлення останнього рейсу має бути після часу прибуття
передостаннього:
            final Flight beforeLast = base.getBeforeLast();
            if (last.getDepartureTime() < beforeLast.getArrivalTime()) {
                // Далі ми рухатися по цьому ланцюжку не можемо - відсікаємо
гілку
                return Collections.emptyList();
            }
            if (last.getTo() == base.getReturnPoint()) { //!перевіряємо чи може бути
циклом
                // Останній рейс у ланцюжку повертається у початкову точку -
цикл знайдено:
                return Arrays.asList(base);
            }
        }
        // Продовжити рекурсивно для всіх сусідніх рейсів:
        return getNeighbours(last)
```

```
                .map(flight -> detectCycles(new Cycle(base, flight))) //!це рекурсія
                .flatMap(List::stream)
                .collect(Collectors.toList());
    }

    private Stream<Flight> getNeighbours(Flight flight) {
        return allFlights.stream().filter(next -> next.getFrom() == flight.getTo());
    }

    private static int getWaitExpenses(int time) {
        final double waitPrice = 0.3; // TODO треба поекспериментувати із
різними коефіцієнтами
        return (int) Math.round(time * waitPrice);
    }

}
package edu.kpi.ipsa.opavloshchuk.airways;
import java.util.List;
import java.util.ArrayList;
import edu.kpi.ipsa.opavloshchuk.airways.calculation.Calculator;
import edu.kpi.ipsa.opavloshchuk.airways.data.Cycle;
import edu.kpi.ipsa.opavloshchuk.airways.data.Flight;
import edu.kpi.ipsa.opavloshchuk.airways.data.FlightValidator;
import edu.kpi.ipsa.opavloshchuk.airways.data.FlightsStorage;
import edu.kpi.ipsa.opavloshchuk.airways.upload.csv.CsvParser;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
@Scope("session")
public class MainController {

    private final FlightsStorage sourceFlightStorage = new FlightsStorage();
    private final List<Cycle> cycles = new ArrayList<>();
    private final List<Flight> mandatoryFlightsWithoutCycles = new
ArrayList<>();
    private final Map<String, String> validationErrors = new HashMap<>();
    private final List<String> importErrors = new ArrayList<>();

    // Відкрити головну сторінку
    @GetMapping("/")
    public String home(Model model) {
        clearErrors();
        return goHome(model);
    }

    @PostMapping("/")
    public String addFlight(@ModelAttribute Flight flight, Model model) {
        clearErrors();
        final Map<String, String> valErrors = new FlightValidator().apply(flight);
```

```
        if (valErrors.isEmpty()) {
            sourceFlightStorage.store(flight);
        } else {
            validationErrors.putAll(valErrors);
        }
        return goHome(model);
    }
     @GetMapping("/remove")
    public String removeFlight(@RequestParam(name = "number", required = true)
int number, Model model) {
        sourceFlightStorage.remove(number);
        clearErrors();
        return goHome(model);
    }
    @PostMapping("/uploadCsv")
    public String uploadCsv(@RequestParam("file") MultipartFile file,
            RedirectAttributes redirectAttributes, Model model) throws IOException {
        cycles.clear();
        sourceFlightStorage.clear();
        mandatoryFlightsWithoutCycles.clear();
        clearErrors();
        final CsvParser parser = new CsvParser(file.getBytes());
        parser.perform();
        parser.getFlights().forEach(flight -> sourceFlightStorage.store(flight));
        importErrors.addAll(parser.getErrors());
        return goHome(model);
    }
    @GetMapping("/calculate")
    public String calculate(Model model) {
        cycles.clear();
```

```
        mandatoryFlightsWithoutCycles.clear();
        clearErrors();
        final Calculator calculator = new Calculator(sourceFlightStorage.list());
        calculator.perform();
        cycles.addAll(calculator.getCycles());

mandatoryFlightsWithoutCycles.addAll(calculator.getMandatoryFlightsWithoutCy
cles());
        return goHome(model);
    }

    private String goHome(Model model) {
        model.addAttribute("flight", new Flight());
        model.addAttribute("source", sourceFlightStorage.list());
        model.addAttribute("cycles", cycles);
        model.addAttribute("mandatoryFlightsWithoutCycles",
mandatoryFlightsWithoutCycles);
        model.addAttribute("validationErrors", validationErrors);
        model.addAttribute("importErrors", importErrors);
        return "home";
    }
    private void clearErrors() {
        importErrors.clear();
        validationErrors.clear();
    }
}
package edu.kpi.ipsa.opavloshchuk.airways.data;
public class Flight {
    private int number; // номер. Унікальний ключ
    private int from; // аеропорт, звідки рейс починається
```

```
private int to; // аеропорт, де рейс закінчується
private int income; // прибуток від рейсу або пріорітет рейсу
private int expenses; // витрати на рейс
private int departureTime; // час відльоту
private int arrivalTime; // час прильоту
private boolean mandatory; // рейс є обов'язковим?
public int getNumber() {
   return number;
}
public void setNumber(int number) {
   this.number = number;
}
public int getFrom() {
   return from;
}
public void setFrom(int from) {
   this.from = from;
}
public int getTo() {
   return to;
}
public void setTo(int to) {
   this.to = to;
}
public int getDepartureTime() {
   return departureTime;
}
public void setDepartureTime(int departureTime) {
   this.departureTime = departureTime;
}
```

```
    public int getArrivalTime() {
        return arrivalTime;
    }
    public void setArrivalTime(int arrivalTime) {
        this.arrivalTime = arrivalTime;
    }
    public boolean isMandatory() {
        return mandatory;
    }
    public void setMandatory(boolean mandatory) {
        this.mandatory = mandatory;
    }
    public int getIncome() {
        return income;
    }
    public void setIncome(int income) {
        this.income = income;
    }
    public int getExpenses() {
        return expenses;
    }
    public void setExpenses(int expenses) {
        this.expenses = expenses;
    }
}
package edu.kpi.ipsa.opavloshchuk.airways.data;
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
```

```
public class FlightsStorage {

    // Внутрішній контейнер рейсів. Забезпечує унікальність рейсу за номером
    private final Map<Integer, Flight> data = new HashMap<>();

    public void store(Flight flight) {
        if (flight != null) {
            data.put(flight.getNumber(), flight);
        }
    }

    public void remove(int number) {
        if (data.containsKey(number)) {
            data.remove(number);
        }
    }

    public List<Flight> list() {
        final List<Flight> result = new ArrayList<>(data.values());
        result.sort((f1, f2) -> f1.getDepartureTime() - f2.getDepartureTime());
        return result;
    }

    public void clear() {
        data.clear();
    }

}
package edu.kpi.ipsa.opavloshchuk.airways.data;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

public class FlightValidator implements Function<Flight, Map<String, String>> {

    @Override
    public Map<String, String> apply(Flight flight) {
        final Map<String, String> result = new HashMap<>();
        // Місце відправлення та місце призначення мають бути різними
        if (flight.getFrom() == flight.getTo()) {
            result.put("to", "Start and end points of flight are the same");
        }
        // Час відправлення має бути меншим, ніж час прибуття
        if (flight.getDepartureTime() > flight.getArrivalTime()) {
            result.put("arrivalTime", "Arrival time is less than departure time");
        }
        return result;
    }

}
<!DOCTYPE html>

<html xmlns:th="https://www.thymeleaf.org"> <!-- мова розмітки, входить в спрінгБут -->
    <head>
        <title>AIRWAYS application</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
        <style type="text/css">
            .error {
                color: red;
            }
            td {
                vertical-align: top;
            }
        </style>

    </head>
    <body>
        <h1>AIRWAYS application:
        <a href="#" th:href="@{/calculate}">Calculate</a></h1>
        <form method="POST" enctype="multipart/form-data"
th:action="@{/uploadCsv}" >
            <input type="file" name="file" /> <input type="submit" value="Upload
CSV file" />
        </form>
        <ul th:if="${!importErrors.isEmpty()}">
            Import errors:
            <li th:each="error : ${importErrors}" th:text="${error}"/>
        </ul>
        <hr/>
        <h2>Source</h2>
        <form action="#" th:action="@{/}" th:object="${flight}" method="POST">
            <table>
                <tr>
                    <th>Number</th>
                    <th>From</th>
                    <th>To</th>
```

```
            <th>Income</th>
            <th>Expenses</th>
            <th>Departure time</th>
            <th>Arrival time</th>
            <th>Mandatory</th>
            <th></th>
        </tr>
        <tr>
            <td>
                <input type="number" th:field="*{number}"/>
            </td>
            <td>
                <input type="number" th:field="*{from}"/>
            </td>
            <td>
                <input type="number" th:field="*{to}"/>
                <div class="error" th:if="${validationErrors.containsKey('to')}"
                    th:text="${validationErrors.get('to')}" />
            </td>
            <td>
                <input type="number" th:field="*{income}"/>
            </td>
            <td>
                <input type="number" th:field="*{expenses}"/>
            </td>
            <td>
                <input type="number" th:field="*{departureTime}"/>
            </td>
            <td>
                <input type="number" th:field="*{arrivalTime}"/>
```

```
                <div class="error"
th:if="${validationErrors.containsKey('arrivalTime')}"
                    th:text="${validationErrors.get('arrivalTime')}" />
            </td>
            <td>
                <select th:field="*{mandatory}">
                    <option value="true" th:value="true">true</option>
                    <option value="false" th:value="false">false</option>
                </select>
            </td>
            <td>
                <input type="submit" value="Add or update flight"/>
            </td>
        </tr>
        <tr th:each="flight : ${source}">
            <td th:text="${flight.number}" />
            <td th:text="${flight.from}" />
            <td th:text="${flight.to}" />
            <td th:text="${flight.income}" />
            <td th:text="${flight.expenses}" />
            <td th:text="${flight.departureTime}" />
            <td th:text="${flight.arrivalTime}" />
            <td th:text="${flight.mandatory}" />
            <td>
                <a href="#"
th:href="@{/remove(number=${flight.number})}">Remove</a>
            </td>
        </tr>
    </table>
</form>
```

```
<div th:if="${!cycles.isEmpty()}">
    <hr/>
    <h2>Detected cycles</h2>
    <table th:each="cycle : ${cycles}">
        <tr>
            <td colspan="7" th:text="${cycle.getTag()}"/>
        </tr>
        <tr>
            <th>Number</th>
            <th>From</th>
            <th>To</th>
            <th>Income</th>
            <th>Expenses</th>
            <th>Departure time</th>
            <th>Arrival time</th>
            <th>Mandatory</th>
        </tr>
        <tr th:each="flight : ${cycle.getFlights()}">
            <td th:text="${flight.number}" />
            <td th:text="${flight.from}" />
            <td th:text="${flight.to}" />
            <td th:text="${flight.income}" />
            <td th:text="${flight.expenses}" />
            <td th:text="${flight.departureTime}" />
            <td th:text="${flight.arrivalTime}" />
            <td th:text="${flight.mandatory}" />
        </tr>
    </table>
</div>
<div th:if="${!mandatoryFlightsWithoutCycles.isEmpty()}">
```

```
        <hr/>
        <h2>Detected mandatory flights without cycles</h2>
        <table>
          <tr>
            <th>Number</th>
            <th>From</th>
            <th>To</th>
            <th>Income</th>
            <th>Expenses</th>
            <th>Departure time</th>
            <th>Arrival time</th>
            <th>Mandatory</th>
          </tr>
          <tr th:each="flight : ${mandatoryFlightsWithoutCycles}">
            <td th:text="${flight.number}" />
            <td th:text="${flight.from}" />
            <td th:text="${flight.to}" />
            <td th:text="${flight.income}" />
            <td th:text="${flight.expenses}" />
            <td th:text="${flight.departureTime}" />
            <td th:text="${flight.arrivalTime}" />
            <td th:text="${flight.mandatory}" />
          </tr>
        </table>
      </div>
    </body>
</html>
```